

السيد : محمود فايد
msfclipper@hotmail.com

اسرع طريق لتعلم البرمجة
Fast Programming way using VFP
Version 6 (common) then 9(the last)

إهداء:

- الى والدتى الحبيبة التى طالما تحملت الالم من اجل سعادتى
- الى كل راغب فى العلم وعاشق للمعرفة

شكر خاص :

- المهندس : سمير ابراهيم فايد
  - اول من علمنى ان اضع اصابعى على لوحة المفاتيح
  - اول من ادخلنى الى عالم البرمجة
- المهندس : مجدى محمد ابو العطا
  - تعلمت من المراجع التى قام بتاليفها ما لم اتعلمه من اى مرجع اخر
- الدكتور المهندس : إيهاب عزيز خليل
  - اول من علمنى كيف يكون البحث العلمى
- الدكتور المهندس : صلاح شعبان
  - هو الذى جعلنى احترف نمط برمجة الكائنات OOP
- السيد : فرنس باناكير
  - اول من علمنى كيفية المشاركة الفعلية فى المشاريع مفتوحة المصدر
- السيد : انتون فان
  - اول من علمنى اساسيات نمط برمجة الكائنات
- المهندس : محمد سمير فايد
  - اول من علمنى اساسيات SQL Server
- الدكتور : فادى سعيد فايد
  - لمساعدته لى فى الحصول على العديد من البرامج
- المهندس : محمد سعود (مهدى)
  - اول من ارشدنى الى VFP ومدى اهميتها
- الدكتور : عمرو طلبة
  - اول من ارشدنى الى نمط برمجة العميل AOP
- المهندس : أحمد الكورانى
  - علمنى اساسيات الالكترونيات
- المهندس : أحمد عبدالعظيم غزيه
  - علمنى اساسيات لغة JAVA
- المهندس : محمد الهدهد
  - كتابه المجانى الجيد فى دروس قواعد بيانات VFP شجعنى على المساهمة فى هذا الاتجاه مع مراعاة عدم التكرار ولذلك انصح بشدة بقراءة هذا الكتاب.

معلومات حول الكاتب :

السيد : مود إبراهيم فايد – مواليد ١٩٨٦|١٢|٢٩ دينة سرس الليان – افظة ا نوفية – هورية مصر العربية - (طالب بكلية ا ندسة الالكترونية دينة منوف)

من الشباب ا دد الذين تربوا منذ نعومة اظفارهم على استخدام ا اسب وبر ته فقد بدا ال ة عام ١٩٩٧ وهو العاشرة من العمر وخلال فترة زمنية قدرها اربعة سنوات كن من عمل اول برنامج كن بيعه الاسواق (حسابات خروج ا رضى – لاحد ا ستشفيات الك ى دينة جدة – ا ملكة العربية السعودية) تلى ذلك العديد من ال امج التجارية

عام ٢٠٠٠ ا ه بفطرته ا التعمق بر ة نظام تشغيل ا اسب (وبالتحديد نظام ادارة البيئة الرسومية) وكان سبب ذلك الا اه الانبهار الشديد بالاختلاف ا ذرى ب نظامى دوس و وندوز .

عام ٢٠٠٥ كن من اصدار نظام تشغيل ادارة البيئة الرسومية لكى يعمل باستقرار وا دير بالذكر انه قبل الوصول لتلك ا رحلة اعادة بناء النظام كاملا اكثر من ٧ مرات.

اكتسب من خ ته بر ة نظام ادارة البيئة الرسومية خ ة جيدة ادت ا ابتكار ط بر ة جديد (ط بر ة ا ادم ا متاز) ليكون بديلا لنمط بر ة الكائنات وياخذ الاعتبار الا اط ت الانشاء (ط بر ة العميل ا وجه – ط بر ة اللغات ا وجه – ط بر ة فصل البيانات)

ومن خلال تلك الرحلة اتقن العديد من اللغات مثل **C,Clipper,VFP & xHarbour** وامتلك خ ة جيدة لغات اخرى مثل **VB,JAVA**

و ا شعر بانه قد وصل ا مرحلة جيدة وجد ان تلك الرحلة كان من ا مكن ان تكون اقصر بكث اذا توفرت الكتب العربية ا يدة ال تغ عن ا راجع الاجنبية ال تتسم بالتكامل ا ادة العلمية والبساطة العرض ولكن ثل بعض الصعوبة للناشئ من الشباب.

و ذا فان هذا الكتيب ا تواضع هو اول لبنة هذا الطريق على ا ستوى الشخصى وهو توف موعة جيدة من الكتب العربية و لا ينفى ذلك تقديرى للمجهودات ا بذولة من قبل الاخوة هذا الا ال والذين قد استفدت منهم استفادة لا كن ان انكرها واسال ا ان زيهم يعا كل خ وا و التوفيق.

فكرة الكتيب :

الوقع بعد ان قمت بنشر كتيب صغ صوص ال ة من غ كود باستخدام DoubleS تلقيت العديد من الرسائل من الاخوة الذين يودون اقتحام عا ال ة ويسالون عن الطريق وكيفية التواجد عا ا مع الوقت.

وكان هذا الكتيب هو ردا على تلك الرسائل والذى ارجو من ا عز وجل ان يكون خ عون م لدخول ال ال ة بل ويساعدهم على تثبيت اقدامهم الال وا ضى و الاحتراف.

لماذا فيجوال فوكس برو :

سبب اختيار تلك اللغة ليس لانا اللغة ال اقوم باستخدامها تطوير برا ى بل على العكس لقد اعتدت على استخدام لغات اخرى ولكن سبب الاختيار هو

١ – هذه اللغة لا تقل السهولة عن اللغات الاخرى مثل فيجوال بيسك الا انا تتميز عنها بانا مع ب السهولة التعلم (كفيجوال بيسك) والسهولة الاستخدام

٢ – هى افضل لغة على الاطلاق كن استخدامها تطوير ال امج التجارية (انظمة قواعد البيانات) وذلك باعتراف شركة مايكروسوفت وال تفخر بذا ا نتج

٣ – مع هذه اللغة ب امكانية تطوير تطبيقات كاملة باستخدام الكود فقط او باستخدام ادوات التصميم

٤ – لا تقل الشان عن لغة ا افا من حيث تطوير بر يات باستخدام ط بر ة الكائنات وهى تفوق فيجوال بيسك ذلك

٥ – تنتمى ا عائلة الدوت النت و نفس الوقت مازال يتم تطويرها (الاصدار ٩ ح الان) وهذا يع انك لن تعا كث ا من مشكلة التنقل ب الاصدارات كما حدث مع م فيجوال بيسك عند الانتقال من الاصدار ٦ ا الدوت نت

٦ – ا اد تلك اللغة نحك فرصة جيدة للتعامل مع لغات عديدة من نفس العائلة Xbase مثل فيجوال اكس بيس بلس بلس و كلي و اكس هاربور وفلاج شب والعديد من لغات ال ة القد ة (بعضها ت دوس مثل كلي) وا تطورة (اكس هاربور لكل انظمة التشغيل ا عروفة)

٧ – لا تشتمل اللغة على اي نوع من القيود من حيث نوعية التطبيقات ال يتم تطويرها فهى تشمل كافة يزات فيجوال بيسك كاى لغة بر ة تعمل ت وندوز.

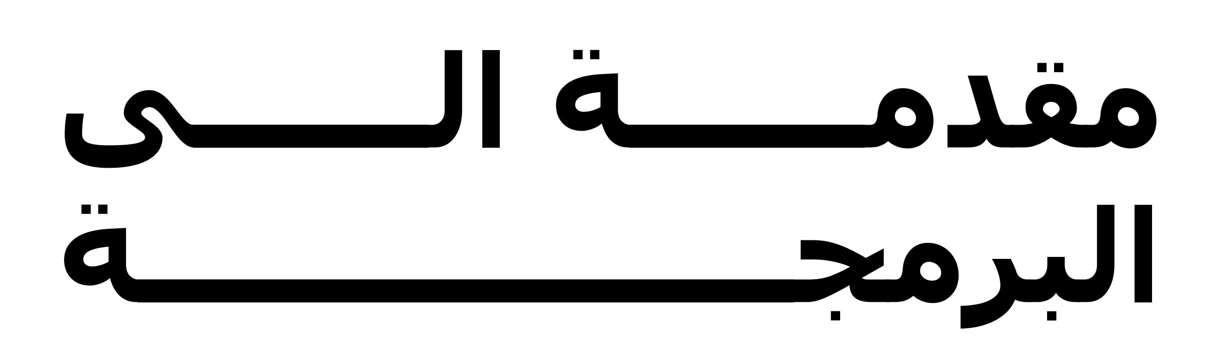
٨ – تتميز اللغة عند تطوير انظمة قواعد البيانات باستخدامها فهى توى على نظام ادارة قواعد بيانات خاص با لذلك لن تاج للخروج من يط تطوير اللغة اثناء العمل

٩ – توى اللغة على عدد هائل من ا عا ات كن استخدامها زيادة سرعة تطوير التطبيقات

لماذا فيجوال فوكس برو نادرة الاستخدام ؟

- يرجع ذلك لسياسة شركة مايكروسوفت بخصوص تسويق ذلك المنتج المتميز
  - فى الواقع لم تعطى شركة مايكروسوفت الاهتمام المطلوب بخصوص تسويق فيجوال فوكس برو والذى يقدر عدد مستخديميها الشرعيين بحوالى ١٠٠،٠٠٠ فقط
  - يرجع السبب فى ذلك الى
    - هذا المنتج كان فى فترة من الاوقات غنى عن التعريف لان تاريخ لغة FoxPro يعود الى عام ١٩٨٤ واشترت حقوقه شركة مايكروسوفت عام ١٩٩٢
    - يحتوى هذا المنتج على نظام ادارة بيانات ولغة برمجة معا فى منتج واحد مما يعنى ان شرائه يضيع على مايكروسوفت شراء منتجين اخرين هما (لغة البرمجة كفيجوال بيسك على سبيل المثال) وقاعدة البيانات SQL SERVER مما يعنى ربح اقل لمايكروسوفت من بيع فيجوال فوكس برو فقط
    - تلقت مايكروسوفت العديد من الطلبات من مئات من المبرمجين بخصوص الاهتمام بتسويق فيجوال فوكس برو ولكن مايكروسوفت لم تهتم بذلك
    - ازداد قلق مبرمجين فيجوال فوكس برو بخصوص مستقبل اللغة عند صدور الاصدار ٧ واعلان مايكروسوفت بعزل فيجوال فوكس برو من Visual Studio
    - زال هذا القلق عند استمرار تطور اللغة وصدور الاصدارين ٨ و ٩
    - الاصدار ٩ يعد نقلة نوعية فى تاريخ فيجوال فوكس برو وبمجرد صدور هذا الاصدار فى عام ٢٠٠٤ ازداد الطلب على فيجوال فوكس برو بمعدل كبير – لكن سياسة مايكروسوفت بخصوص تسويق هذا المنتج مازالت كما هى.
    - من الاسباب التى ادت الى انتشار فيجوال فوكس برو موخرا القضية التى تم اثارتها عندما قام عدد من المبرمجين بتشغيل فيجوال فوكس برو من خلال Linux/WINE والذى يعد انتهاك لحقوق مايكروسوفت والتى تشترط ان تعمل البرامج التى يتم تطويرها بلغة فيجوال فوكس برو داخل منصة وندوز .
    - لا ينبغى القلق بخصوص عدم انتشار فيجوال فوكس برو الانتشار الكافى والمطلوب – لانها الاتجاه الصحيح وينبغى سلوكه وان قل عدد الافراد الذين يسيرون فى هذا الطريق – لان الطريق الصحيح يتضح وينتشر مع الوقت.

الباب الأول



مقدمة الى البرمجة :

س : ما هى انواع السوفت وير SoftWare ؟
جـ : ينقسم السوفت وير الى نوعين نظام Software System و Software Program.

س : ما المقصود ب Software System ؟
جـ : هى نوعية السوفت وير التى لا يمكن برمجتها بصورة مباشرة بدون اللجوء الى تصميم مسبق وغالبا ما يكون هذا التصميم غير متوفر فى البداية وتنشا الحاجة لاستخدام علم الذكاء الاصطناعى حتى يمكن الحصول على تصميم حسابى او ما يشبه الالجورزم يمكن استخدام لبرمجة النظام ومن امثلة ذلك نظام ادارة البيئة الرسومية الخاص بنظام تشغيل الحاسب – وعملية تصميم النظم تخضع لعمليات التجربة والخطا حتى الحصول على تصميم مستقر .

س : ما هو علم الذكاء الاصطناعى ؟
جـ : هو علم يهدف لوضع نموذج حسابى او الجورزم لنظم السوفت وير والتى تحتاج الى الذكاء اكثر من العلم لبرمجتها – وفى نفس الوقت هو العلم الذى يهدف لوضع نموذج حسابى او الجورزم للتطبيقات التى تقوم بعمل محاكاة لذكاء الانسان مثل لعبة الشطرنج.

س : ما هو برنامج السوفت وير Software Program ؟
جـ : هى البرامج بصفة عامة غير النظم والتى ترجع عملية برمجتها على خبرة وعلم اكثر من الذكاء ولا تتدخل فى انظمة الذكاء الاصطناعى ومن امثلة ذلك برامج قواعد البيانات والالعاب وغيرها.

س : ماهى انواع التطبيقات الشائعة فى الوطن العربى ؟
جـ : ان برمجة انظمة قواعد البيانات والبرامج التعليمية وقليل من برامج التحكم بالحاسب (واجهة بالحاسب لمتابعة مصنع – سنترال وغيرها) هى البرامح الشائعة وللاسف لا نجد مكانا فى سوق انظمة التشغيل او لغات البرمجة او حتى الادوات الاضافية للغات والسبب فى ذلك (قلة العلماء – قلة مجموعات المبرمجين – سوء الادارة)

س : ما هى مواصفات المبرمج الذى يجيد تطوير انظمة قواعد البيانات ؟
جـ : اولا : القدرة على انشاء قاعدة البيانات والتعامل معها واستخلاص البيانات منها والحصول على التقارير الازمة
ثانياً : القدرة على عمل واجهة جيدة للمستخدم
ثالثا : القدرة على وضع تصميم جيد للبرنامج من خلال نمط برمجة الكائنات
رابعا : القدرة على تصميم انظمة قواعد البيانات باسلوب الزبون – الخادم
خامسا : القدرة على الاستفادة من امكانيات نظام التشغيل وبيئة العمل والتعامل مع الاجهزة الملحقة بالحاسب وغيرها من المهارات التى تميز بين المبرمجين
سادسا : القدرة على الابداع وحل المشاكل الطارئة باحسن السبل

س : ماهو المعدل الزمنى الكافى لتعلم البرمجة ؟
جـ : ثلاثة شهور كافية للحصول على العلم الكافى وذلك بمعدل من ٤ – ٨ ساعات يوميا لشخص الذى يمتلك عقلية رياضية جيدة تجيد الفهم والاستيعاب السريع
و ستة شهور كافية لحفظ ما تم فهمه و ذلك يضمن سرعة العمل من المبرمج وعدم الحاجة الى المراجع فى المسائل البسيطة
وثلاثة شهور اخرى لعمل مشروع متكامل يكفى لان يظهر المبرمج فيه كل مواهبه
وسنة كاملة لاكتساب خبرة السوق والتعامل مع الافراد (مبرمجين زملاء – عملاء)
اى ان عامين فترة جيدة للحصول على مبرمج محترف فى تطوير انظمة قواعد البيانات

س : ما هو المقصود بنظام تشغيل الحاسب ؟
جـ : هو البرنامج الاساسى لادارة مكونات الحاسب ويتكون من
١ – النواة والتى تكون مسئولة عن ادارة المكونات المادية للحاسب وتتكون هذه النواة من نظام لادارة العمليات ونظام لادارة الذاكرة ونظام لادراة الملفات
٢ – واجهة النظام وقد تكون نصية او رسومية
٣ – اذا كانت الواجهة رسومية فانها تتكون من مكتبية جرافك للتعرف على كارت الشاشة والرسم عليه ونظام لادارة البيئة الرسومية والتعامل معها

س : ما المقصود بنظام ادارة الاحداث ؟
جـ : هو نظام يجعل البرنامج قادر على الاستجابة لطلبات المستخدم من خلال وحدات الادخال وفى نفس الوقت يقوم بعمل عمليات اخرى

س : ما المقصود بتعدد المهام ؟
جـ : هى قابلية البرنامج لعمل اكثر من عملية فى وقت واحد (الايحاء بذلك من خلال المعدل الزمنى حيث ان سرعة الحاسب كبيرة)

س : ما هى عملية البرمجة ؟
جـ : هى عملية يقصد من خلالها تخزين تعليمات للحاسب الالى لكى يقوم بوظيفة محددة تخدم غرض معين قد يكون غرض مباشر مثل البرامج التى تخدم المستخدم مباشرة او غرض غير مباشر مثل تطوير ادوات تساعد المبرمجين على اداء عملهم.

س : كيف يمكن برمجة الحاسب ؟
جـ : يتم ذلك باستخدام لغة برمجة تكون وسيط بين الشخص الذى يقوم ببرمجة الحاسب وبين لغة الالة التى يفهمها الحاسب

س : مما تتكون لغة البرمجة ؟
جـ : ١ – على الاقل على مترجم يقوم بتحويل البرنامج الذى يتم كتابته الى لغة الالة بعد التاكد من خلوه من الاخطاء
٢ – تحتوى اللغات المتطورة على العديد من الادوات لزيادة سرعة عمل التطبيقات

س : ماهى اجيال لغات البرمجة ؟
جـ : هناك ٥ اجيال هى
١ – لغة الالة
٢ – لغة التجميع
٣ – لغات عالية المستوى
٤ – لغات متخصصة فى قواعد البيانات
٥ – اللغات الطبيعية

س : ما هى مراحل عمل البرنامج ؟
جـ : ١ – تصميم البرنامج لتحديد الوظائف المطلوبة منه
٢ – اختيار لغة برمجة لعمل البرنامج بها
٣ – كتابة البرنامج
٤ – اختبار البرنامج
٥ – اصلاح الاخطاء

س : ما هى مكونات البرنامج ؟
جـ : يتكون البرنامج من
١ – واجهة للمستخدم يتعامل معها
٢ – ملفات البيانات التى يقوم البرنامج بتخزينها
٣ – التعليمات او الكود

س : ما المقصود بنمط البرمجة ؟
جـ : هو الاسلوب المتبع لتنظيم عملية كتابة التعليمات او الكود داخل البرنامج
١ – طريقة سقوط المياه water fall method
٢ – طريقة البرمجة الهيكلية structure programming
٣ – طريقة برمجة الكائنات object oriented programming (OOP)
٤ – طريقة الخادم الممتاز Super Server (DoubleS)

س : ما هى انواع البرمجيات التى يتم تطويرها ؟

جـ : ١ – انظمة التشغيل
٢ – برمجيات النظام
٣ – لغات البرمجة
٤ – ادوات ومكتبات للغات
٥ – انظمة قواعد البيانات
٦ – الالعاب
٧ – البرامج التعليمية
٨ – برامج الاتصالات والشبكات
٩ – برامج معالجة الصوت (كبائن المكالمات)
١٠- برامج واجهات التحكم (التحكم بالمصانع)
١١- البرامج الرياضية والهندسية
١٢ – البرامج المكتبية
١٣ – برامج التصميم والجرافيك

س : ماهى انماط عمل شاشة الحاسب ؟

جـ : يوجد نمطين نمط نصى text mode ويستخدم لعرض الحروف والعلامات والرموز فقط ولا يمكن استخدامه للرسم على الشاشة ونمط اخر رسومى graphic mode يمكن ان يستخدم فى عرض الصور

س : كيف يتم تصنيف واجهة البرنامج ؟

جـ : هناك ثلاث انواع
١ – واجهة خطية يتم استخدام لوحة المفاتيح فقط للتعامل مع البرنامج مثل واجهة نظام التشغيل دوس
٢ – واجهة بالقوائم والتى يتم فيها استخدام الاسهم لاختيار عنصر داخل قائمة
٣ – واجهة رسومية ويتم فيها استخدام الفارة والايقونات للتحكم فى عمل البرنامج

س : ما هى خطوات تعلم لغة البرمجة ؟

جـ : ١ – تحميل لغة البرمجة على حاسبك الشخصى
٢ – تعلم كيفية تشغيلها واغلاقها
٣ – تعلم كيف تتعامل مع واجهة اللغة (انشاء وحفظ وفتح المشاريع)
٤ – فهم العناصر التى يحتوى عليها المشروع وماذا يمثل كل عنصر فى البرنامج
٥ – تعلم اين تصمم شاشات برامجك واين تكتب الاكواد الخاصة بها
٦ – تعلم مكونات الكود (هياكل البيانات – تركيبات التحكم – نمط البرمجة – وظائف اللغة)
٧ – تعلم كيفية الوصول الى التعليمات من خلال شاشات المساعدة HELP
٨ – تعلم كيف تصمم انظمة قواعد البيانات
٩ – تعلم كيف تنشى ملفات البيانات وتقوم بتعديلها
١٠ – كيفية التعامل مع ملفات البيانات من خلال الكود
١١ – تعلم كيفية استخدام ادوات اللغة Extensions and tools
١٢ – كيفية الاستفادة من النظام واستخدام ادواته Windows API
١٣ – كيف تصحح اخطاء برامجك وتختبرها
١٤ – الحصول على حلول مشاكل محددة عبر الانترنت
١٥ – تعلم برمجة الكائنات OOP
١٦ – تعلم تصميم وبرمجة قواعد البيانات من خلال تقنية الزبوم- الخادم Client –Server

ان القدرة على كتابة الاكواد لحل المشاكل المختلفة والحصول على المعلومات المطلوبة من خلال شاشات المساعدة او الانترنت هى العنصر الاساسى للاستدلال على مهارتك فى البرمجة

باختصار لتعلم لغة البرمجة ينبغة عليك الالمام بالمفاهيم الاساسية للبرمجة والقدرة على التعامل مع محيط التطوير الخاص باللغة والالمام بقواعد اللغة لكتابة الكود والبراعة فى الحصول على المعلومات التى تحتاجها من خلال الكتب الالكترونية او الانترنت.

ويزداد الطلب على المفاهيم التى ينبغة الالمام بها باختلاف نوعية البرامج التى تقوم بتطويرها

والان دعنا نظر الى نظام التشغيل وندوز WINDOWS ولكن بعيون اخرى عيون تبحث عن تطوير تطبيقات تعمل تحت هذا النظام وتشدو نحو الاستفادة القصوى منه



شكل ١ – نظام التشغيل WINDOWS

من خلال شكل ١ نرى صورة عادية جدا ومالوفة بالنسبة لنا داخل WINDOWS الا وهى صورة سطح المكتب الملىء بالايقونات المختلفة ويوجد اسفل الصورة شريط المهام الذى يحتوى على قائمة ابدا المعتادة بالنسبة لنا.

ولكن نصل الى المفهوم المطلوب دعنا نرى صورة اخرى لنظام تشغيل اخر قديم جدا يدعى DOS

```
C:\>DIR /W
 Volume in drive C has no label.
 Volume Serial Number is 435E-B791

 Directory of C:\

[WINDOWS]                [Documents and Settings] [Program Files]
[Inetpub]                CONFIG.SYS               AUTOEXEC.BAT
[My Download Files]      csb.log                  DOCUMENTS
[MYSW1]                  [lab1]                   [labsetup]
[MAHLAB1]                [na1]                    [com]
[mcomsetup]              [MCOM1]                  [clipper5]
[BC5]                    [C700]                   config.old
autoexec.old             [djgpp]                  [cyber1]
AUTOEXEC.BA1             [myccode]                [Mahmoudcyber]
[DoubleS]                [DOUBLEStest]            [DoubleS2006]
ahmed.txt                [mydoscom]               damp.zip
[damp]                   [OPENPATH]               [allegro]
[xharbour]               [CLIP]                   [fgl_v30]
[My Music]
               9 File(s)        377,348 bytes
              31 Dir(s)   1,185,632,256 bytes free

C:\>_
```

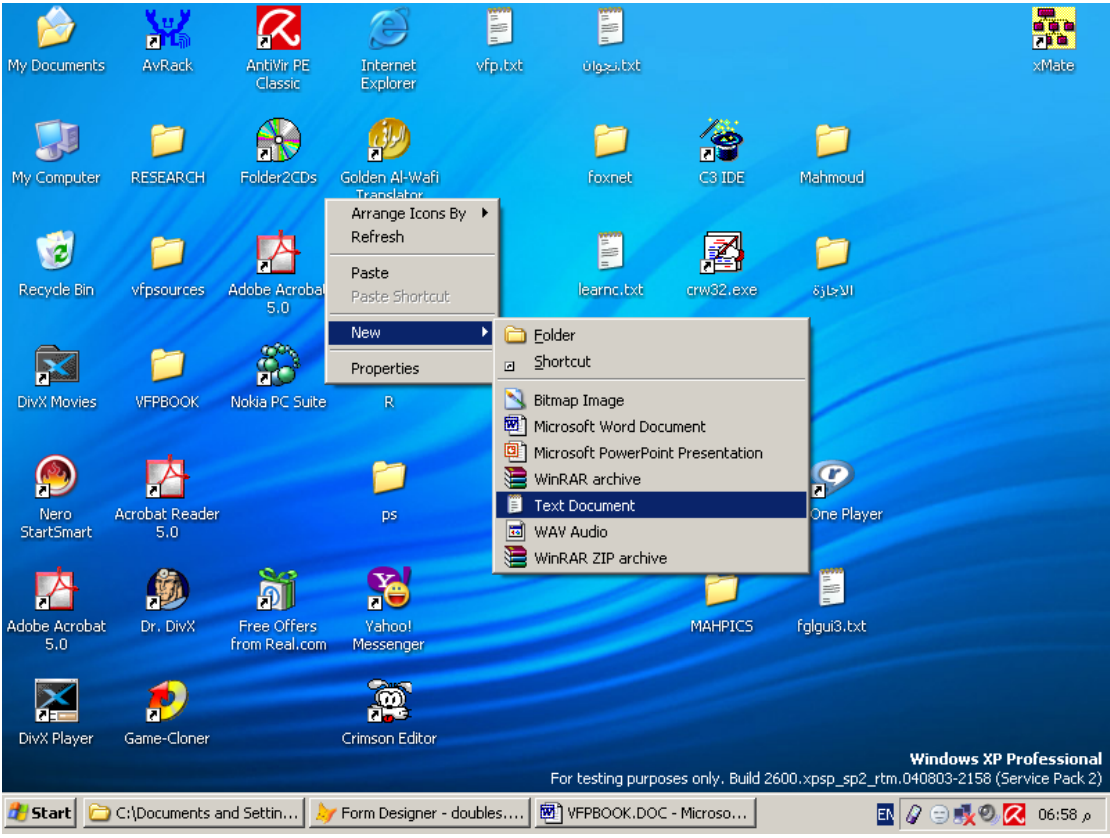
شكل ٢ : نظام التشغيل DOS

ان المستخدم العادى يمكن ان يلاحظ الفرق المباشر بين النظامين وهو ان نظام WINDOWS يعمل فى نمط رسومى Graphic mode بينما النظام DOS يعمل فى نمط نصى Text Mode

ولكن الفرق بينهما فى الواقع اكبر من ذلك بكثير ان نظام Windows يعمل بتقنية نظام ادارة الاحداث Event driven system بمعنى انه يمكنه استقبال التعليمات من المستخدم فى اى وقت سواء من الفارة او لوحة المفاتيح بينما يتابع مهام اخرى حيث ان نظام windows لا يعرف ما سوف يفعله المستخدم بالتحديد بل يفرض عدة احتمالات ويتنظر تحقق اى منها فهو كالخادم الملح الذى يعرض كافة الطلبات على الزبون وينتظر كلمة نعم فهو مثلا يقول

تشرب شاى – قهوة – اجيبلك تاكل – طب تتغدى ايه – طب كبياية لين ويفضل كده يعيد ويزيد فى العرض لحد اما الزبون يقول عايز ايه

ال windows يعرض خدماته باستمرار ويمشى وراك فين ما تروح ويعرض طلباته فمثلا لو رحت بالماوس على الايقونة يعمل حسابه انك ممكن تضغط عليها ولو ضغط مثلا بالزر الايمن للفارة فى اى مكان على سطح المكتب بعيدا عن الايقونات يظهر لك شكل ٣ كالاتى



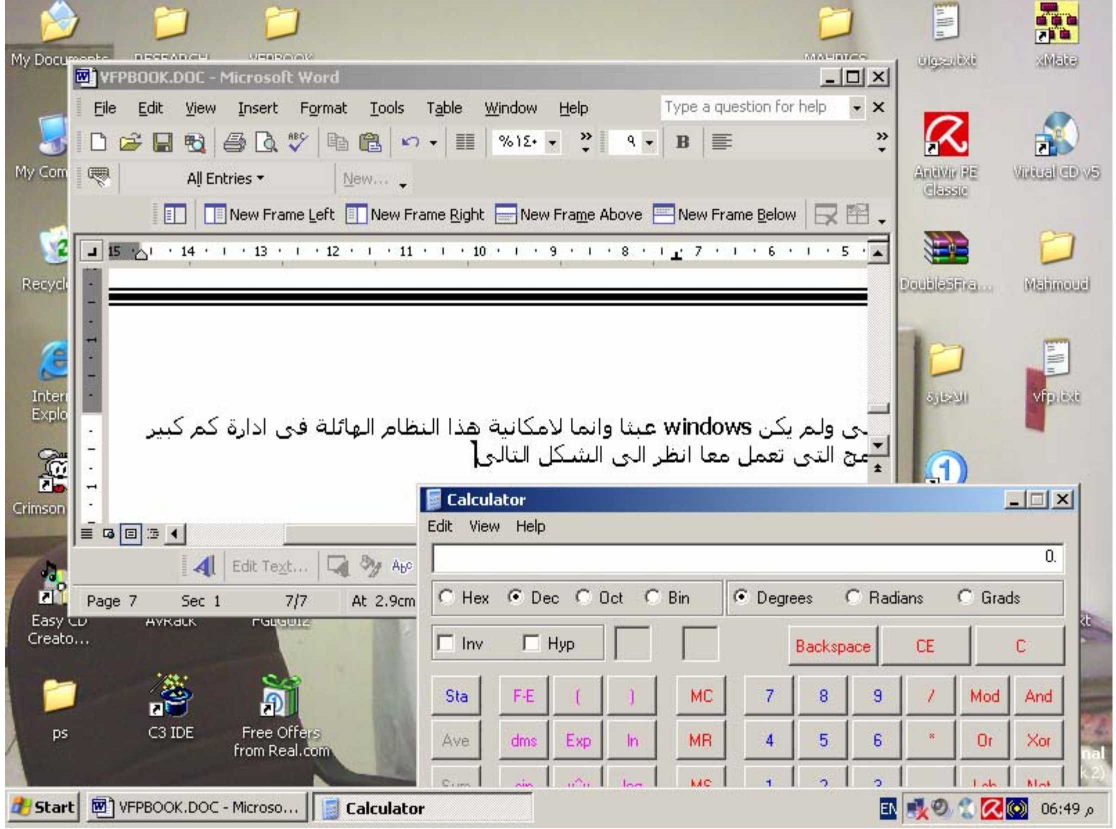
شكل ٣ : ال windows جاهز لاستقبال اى احتمال من خلال نظام ادارة الاحداث

هنا الاحتمالات زادت بالنسبة لل windows حيث انه قد يقوم المستخدم باختيار عنصر من عناصر القائمة الثانية او يعود للقائمة الاولى او قد يذهب بعيد بالفارة ليقف على start وتظهر رسالة click here to begin او قد يستخدم لوحة المفاتيح او ......او......الخ

ومن ثم وحيث اننا سنقوم للبرمجة داخل هذا النظام العملاق فان برامجنا تاخذ نفس ملامحه وحتى الان نعلم ان برامجنا سوف تكون فى البيئة الرسومية وتتبع نظام ادارة الاحداث.

والان دعنا نرى ما يمنحنا هذا النظام من مميزات اخرى !

حقا ان ال windows اسم على مسمى ولم يكن windows عبثا وانما لامكانية هذا النظام الهائلة فى ادارة كم كبير من النوافذ التى تمثل العديد من البرامج التى تعمل معا انظر الى الشكل التالى



شكل ٤ : تعدد النوافذ فى windows

هنا نلاحظ امكانية تشغيل اكثر من برنامج وامكانية windows وقدرته على التعامل مع اكثر من نافذة

اى ان برامجنا التى نقوم بتطويرها تحت windows تستفيد من امكانية تعدد المهام سواء بتشغيل اكثر من برنامج وتعدد النوافذ داخل النظام يسمح لنا باحتواء البرنامج الواحد على اكثر من نافذة فى نفس الوقت

س : ما الفرق بين نظام ادارة الاحداث ونظام تعدد المهام ؟
جـ : نظام ادارة الاحداث يسمح للبرنامج الواحد بعمل اكثر من شى فى نفس الوقت ولكن نظام تعدد المهام يسمح بتشغيل اكثر من برنامج مستقل فى نفس الوقت وهذا وما اردت ان اوضحه من خلال تلك الجولة

س : هل من الضرورى ان ياخذ البرنامج الذى اقوم بتطويره تحت windows شكل التافذة ؟
جـ : من الضرورى ان يكون هناك نافذة ولكن لا يشترط ان تظهر بشكلها المالوف ولهذا فان شكل البرنامج يسمى نموذج FORM يمكن التحكم فى خواصه ليظهر بشكل النوافذ المالوف او باى شكل اخر او يظهر فى ملىء الشاشة.



شكل ٥ : برنامج يعمل فى ملىء الشاشة تحت WINDOWS

ملحوظة :
نظرا لما تحتويه البيئة الرسومية من امكانيات هائلة جعلت من اصعب على المبرمجين تصميم شاشات برامجهم مباشرة باستخدام التعليمات او الكود ولهذا توفرت الادوات التى تسعل عملية تصميم الاشكال او النماذج FORMS من خلال برنامج يقوم بعملية التصميم يسمى FORM DESIGNER مصميم التماذج
وفى اللغات السابقة كانت تلك البرامج توزع بصورة منفصلة عن لغة البرمجة اما الان وخصوصا تحت بيئة WINDOWS فقد اصبح مصمم النماذج جزء لا يتجزاء من لغة البرمجة وهو لا يسمح فقط بتصميم شكل البرنامج بل يتيح لك الفرصة لكتابة الاكواد او التعليمات المرتبطة به داخل النموذج حيث ان مصمم التماذج يضع فى الاعتبار نظام ادالاة الاحداث المتوفر لبرنامجك ويتيح لك كتابة الاكواد المرتبطة بتلك الاحداث

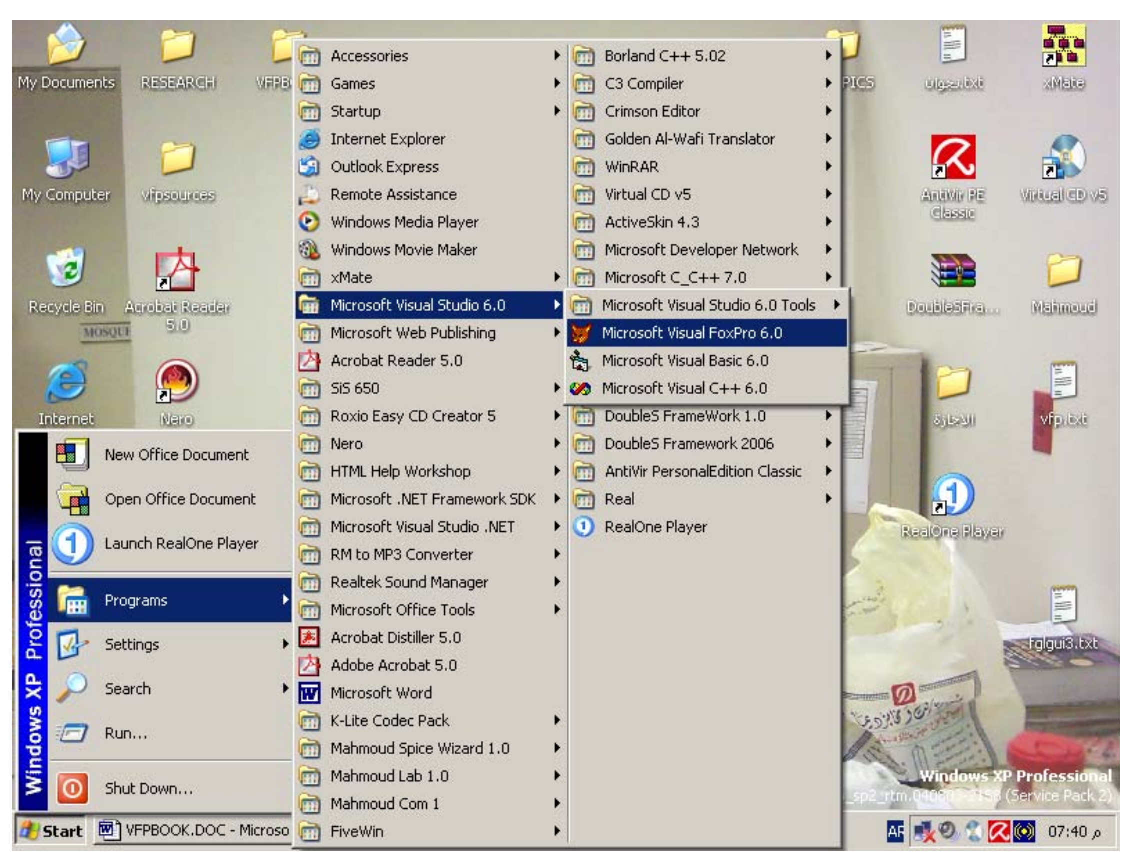
اى ان المبرمج تحت WINDOWS يقوم بتصميم برامجه بالفارة من خلال مصمم النماذج ويتحكم بالبرنامج من خلال كتابة الاكواد التى ترتبط بالاحداث او الاحتمالات .

تاريخ فيجوال فوكس برو :-

يرجع تاريخ تلك اللغة الى امد بعيد – فى بداية الثمانيان مع انتشار اجهزة الحاسب الشخصى وتتطورها تم عمل نسخة من قاعدة البيانات +Dbase III لتعمل تحت نظام DOS ونظرا لنجاح هذا البرنامج بدات العديد من الشركات المنافسة بانتاج برامج شبيهة لتنافس تلك اللغة وكان من هذه البرامج واهمها Clipper, FoxBase & Quick Silver وكان يحتل المرتبة الاولى كليبر Clipper والتى تعتبر لغة برمجة لاحتوائها على مترجم Compiler ينتج ملفات exe ونشا برنامج كليبر عام 1984 وبدا فى الانتشار بالاصدار clipper 87 واستمر فى التطور حتى Clipper 5 ثم ظهر وندوز لينتهى برنامج كليبر بالاصدارين 5.3b , CA-Clipper 5.2e وفى الجانب الاخر تطورت FoxBase والتى كانت تتميز بالسرعة الى + FoxBASE ثم FoxPro واستمرت حتى FoxPro 2.6 for Dos ثم جاءت FoxPro For Windows واستمرت حتى Visual FoxPro 3.0 والذى كان نقلة نوعية للتحول به الفوكس برو من مجرد قاعدة بيانات الى لغة برمجة ثم ظهرت Visual FoxPro 5 بدون المرور بالاصدار 4 ثم 6,7,8,9

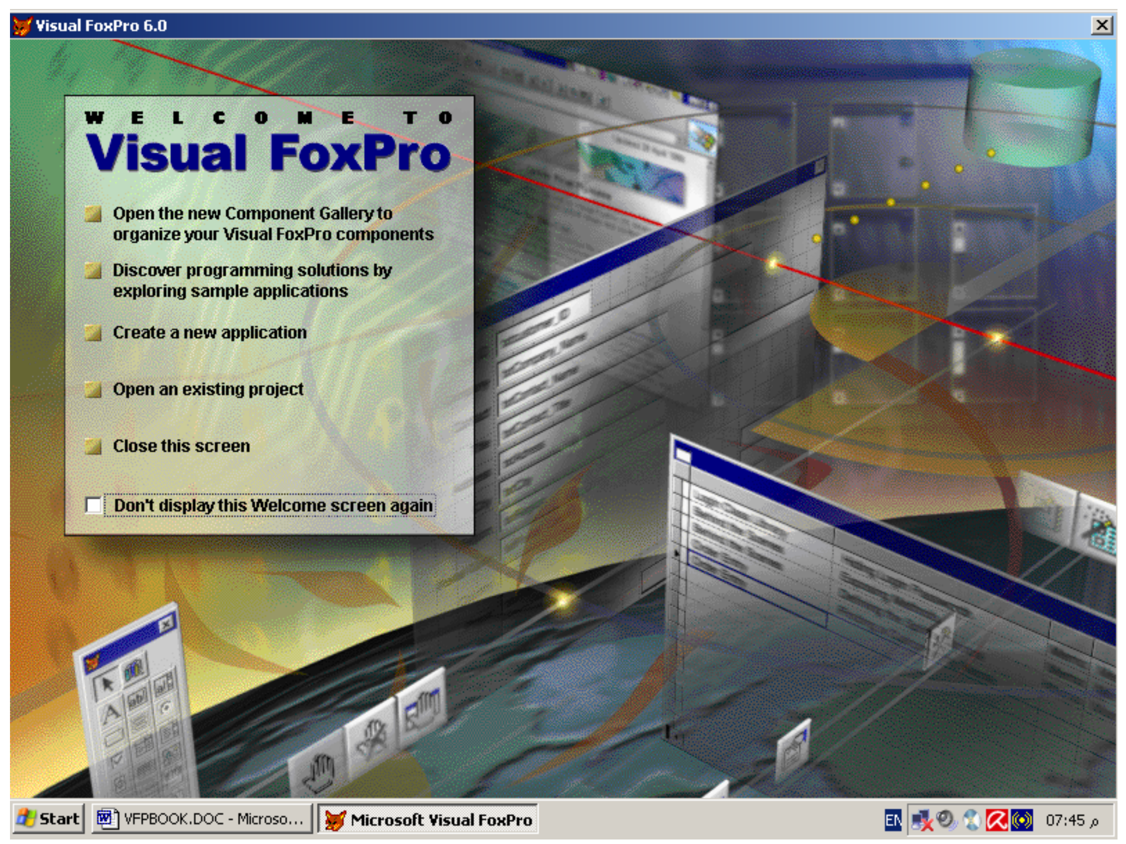
تشغيل فيجوال فوكس برو ٦ :

من خلال المجموعة البرمجية الخاصة بها
١ – اضغط start
٢ – اختار programs
٣ – اختار Microsoft visual studio 6.0
٣ – اختار Microsoft Visual FoxPro 6.0

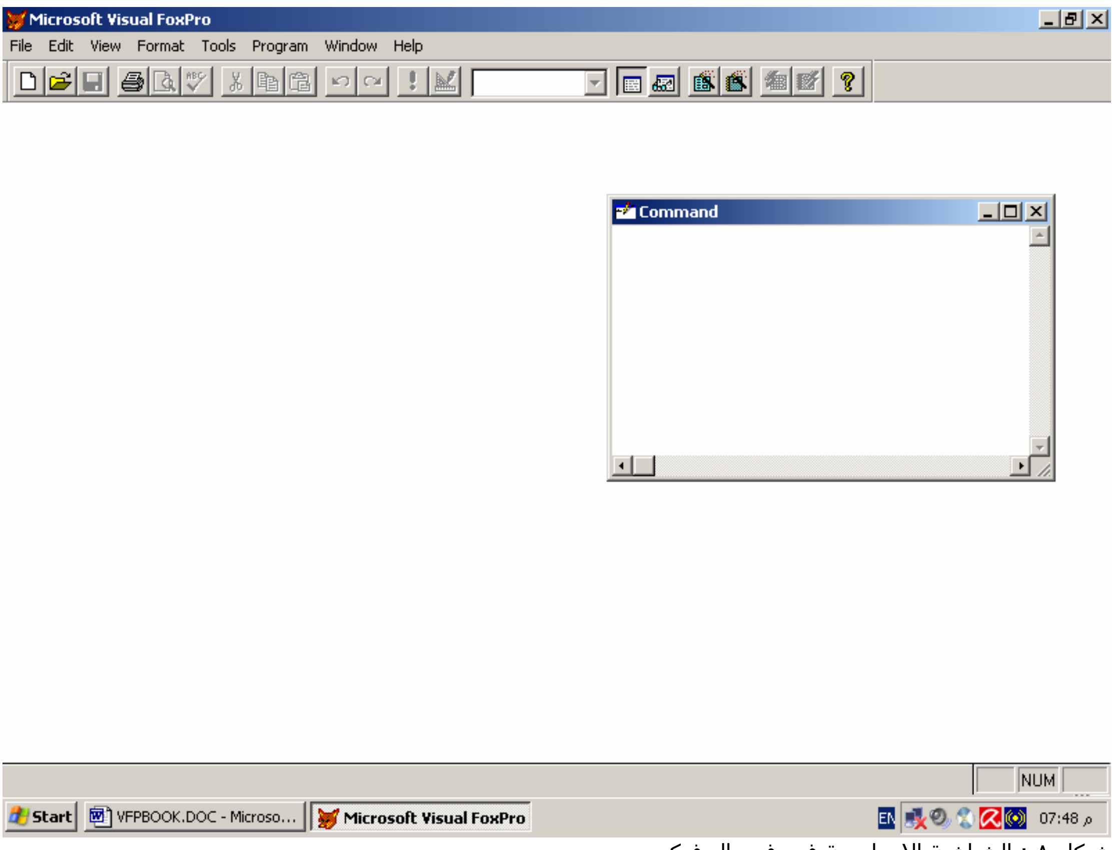


شكل ٦ : تشغيل فيجوال فوكس برو ٦

سوف تظهر شاشة تتيح لك عدة خيارات سوف يتم دراستها فيما بعد ولكن الان اختار close this screen لكى تظر لك شاشة اخرى تحتوى على نافذة تسمى command window



شكل ٧ : شاشة البداية لفيجوال فوكس برو



شكل ٨ : الشاشة الاساسية فى فيجوال فوكس برو

نافذة الامر (Command Window) :

من خلال هذه النافذة المثيرة يمكنك ادخال اوامر لغة فيجوال فوكس برو وتنفيذها مباشرة ويمكنك تنفيذ امر امر من خلالها وذلك بكتابة الامر ثم ضغط مفتاح الادخال (enter) او تنفيذ دفعة اوامر (تعليمات) مرة واحدة.

وفى حالة ادخال امر خطا سوف تحصل على رسالة تفيد بذلك كما فى الشكل التالى :



شكل ٩ : نتيجة ادخال امر خطا فى نافذة الاوامر

ومن المتوقع الحصول على هذا الخطا لان فيجوال فوكس برو لا تحتوى على امر يدعى Hello
وكمثال لاحد اوامر فيجوال فوكس برو الامر

? "كم انا سعيد بتعلم البرمجة" font "" , 20

وهذا الامر يقوم بطباعة رسالة على الشاشة بخط معين (لم نحدد الخط هنا) وحجم خط محدد (٢٠)



شكل ١٠ : عرض رسالة على الشاشة من خلال نافذة الاوامر

ويمكن تنفيذ مجموعة من التعليمات معا مرة واحدة وذلك بكاتبة التعليمات ثم ضغط الزر الايمن للفارة لتظهر قائمة الاختصارات ونختار منها Execute Selection



شكل ١١ : تنفيذ مجموعة من التعليمات دفعة واحدة

والتعليمات التى تم تنفيذها هى

```
SET COLOR TO W/B
CLEAR
? "تم مسح الشاشة بعد اختيار اللون الازرق للخلفية والابيض للنص" FONT "",20
```

تم مسح الشاشة بعد اختيار اللون الازرق للخلفية والابيض للنص للنص

شكل ١٢ : نتيجة تنفيذ التعليمات السابقة معا دفعة واحدة

ومن هنا قد تعرضنا لامرين جديدين احدهما هو الامر CLEAR والذى يفهم من معناه انه يقوم بمسح الشاشة وذلك بالالوان المحددة
والامر الثانى هو الامر SET COLOR TO والذى استخدم لتحديد الالوان وياخذ ذلك الامر لون الكتابة ثم العلامة المائلة / يليها بعد ذلك لون الخلفية
والجدول التالى يبين الالوان التى يتعامل معها هذا الامر

Color	Code
Black	N
Blank	X
Blue	B
Brown	GR
Cyan	BG
Green	G
Inverse	I
Magenta	RB
Red	R
White	W
Yellow	GR+
Underlined	U

جدول ١ : جدول الرموز التى تعبر عن الالوان

والان وقد اجدنا كيفية تنفيذ مجموعة من التعليمات معا سوف نستخدم تلك الطريقة البسيطة والسريعة فى تنفيذ العديد من الامثلة القادمة والتى سوف تدربك على ملامح لغة فيجوال فوكس برو.

الدوال او الوظائف (Functions) :

هى احد اهم مكونات لغة البرمجة وهى الوسيلة السريعة لتنفيذ العديد من المهام فكما فى علم الرياضيات عرفنا مفهوم الدوال (كالدالة التربيعية والدالة اللوغارتيمية وغيرها) فان هذا المفهوم انتقل الى علم البرمجة بصورة مشابهة تماما حيث تحتوى لغة البرمجة على العديد من الوظائف التى لها اسم محدد وتاخذ بيانات(معطيات) لتعمل عليها وتعطى نتيجة محددة
اى انه للتعامل مع الوظيفة او الدالة ينبغى معرفة اسمها وفائدتها والمعطيات التى تستقبلها والقيم التى تعيدها
FUNCTION_NAME(PARA1,PARA2,PARA3,.........) => VALUE
بعض الدوال لا تاخذ معطيات وبعضها لايرجع قيم وهنا دوال لاتاخذ معطيات ولا ترجع قيم (فى هذه الحالة يفضل مصممى لغة البرمجة وضع هذه الدالة على صورة امر COMMAND)
اذا كانت الدالة ترجع قيمة فانه يمكن تخزين هذه القيمة للتعامل معها ويتم ذلك من خلال المتغيرات

مفهوم البرمجة الهيكلية\ التركيبية(Structure Programming) :

ينص هذا المفهوم على ان البرنامج مكون من مجموعة من الوظائف تنادى بعضها البعض لتودى الهدف المنشود من البرنامج – ومن البديهى ان يحتوى البرنامج على نقطة بداية من خلال الدلة الرئيسية فى البرنامج ونظرا لان البرنامج قد يحتوى على العديد من الوظائف فانه يطلق على البرنامج اسم الاجراء Procedure ومن هنا فان البرنامج قد يحتوى على مجموعة من Procedures التى تنادى بعضها البعض بحيث ان كل Procedure يحتوى على دالة او مجموعة من الدوال
الخلاصة :
١ - يحتوى البرنامح على واحد او مجموعة من الملفات (الاجراءات) Procedures
٢ – يحتوى الاجراء Procedure على واحد او مجموعة من الدوال Functions والتعليمات او الاوامر command
٣ – قد تنادى الدالة اثناء عملها دالة اخرى
٤ – اثناء نداء اجراء او دالة لاجراء اخر او دالة اخرى فانه بعد انتهاء تنفيذ الاجراء او الدالة الفرعية فان التنفيذ يعود للدالة الاصلية ليتم متابعة تنفيذ التعليمات
٥ – الفرق بين الاجراء والدالة
١ – الاجراء قد يحتوى على مجموعة من الدوال
٢ – الدالة ترجع قيمة والاجراء لا يرجع قيمة
٦ – بعض اللغات مثل فيجوال فوكس برو تتيح امكانية تعريف اجراءات فرعية داخل اجراء (ملف) رئيسى وهنا يشبه تعريف الاجراء نفس تعريف الدالة اى انه لتعريف الاجراء هناك طريقتين
* من خلال انشاء ملف اجراء
* من خلال اعلانه داخل ملف اجراء اخر
٧ – يسمح بتداخل الدوال فى نفس التعليمة الواحد او السطر الواحد من البرنامج
Func1(func2(func3(func4())))
هنا سوف يتم تنفيذ الدالة ()func4 واسناد القيمة الناتجة لتكون المعطى الذى يتم ادخاله للدالة ()func3 التى يتم تنفيذها هى الاخرى واسناد القيمة الناتجة لتكون هى المعطى الذى يتم ادخاله للدالة ()func2 والتى يتم تنفيذها ايضا لتكون القيمة الناتجة هى المعطى التى يتم اسناده للدلة ()func1
٨ – عند اعطاء اكثر من مدخل \ معطى للدالة فانه يتم الفصل بينهما باستخدام الفاصلة , كالتالى
Func(para1,para2,para3,para4)
٩ – المعطيات قد تكون اسماء متغيرات او عبارات حرفية او قيم رقمية او غيرها بمعنى اما وضع اسم المتغير او كتابة قيمة مباشرة
١٠ – هناك مفاهيم اكثر فى هذا النحو مثل المؤشرات وغيرها سوف يتم الاشارة اليها فيما بعد

المتغيرات (Variables) :

المتغيرات هى اماكن للتخزين الموقت داخل ذاكرة الحاسب وكمثال لذلك من ارض الواقع حينما تطلب منك والدتك او زوجتك ان تنبهها كمان نصف ساعة لكى تطفى النار على الطعام (الان ادوات الطهى المتطورة تحتوى على منبه داخلى Alarm) فانك تقوم بتخزين هذه المعلومة وهى الطلب الذى طلب منك فى ذاكرة وتعمل على هذا الطلب بمعنى انك قد تنظر للساعة مرة او اكثر قبل ان يحين الوقت وبمجرد وصول ساعة الصفر وتنبيه والدتك بالمعاد المحدد ومرور نصف ساعة اخرى فانك قد تنسى ان والدتك قد طلبت منك شيئا اصلا وذلك لان حاجتك للمعلومة لم تعد متوفرة
بالمثل فان الحاسب يقوم بتخزين المعلومات موقتا فى الذاكرة العشوائية Ram وبمجرد الانتهاء من الحاجة لهذه المعلومات فانه يقوم بمسحها اى انه
١ – يقوم الحاسب بالتخزين الموقت للمعلومات داخل الذاكرة العشوائية
٢ – تحمل هذه المعلومات كل على حدة اسم يطلق عليه اسم المتغير مما يسهل الوصول لهذه المعلومات
٣ – بمجرد انتهاء الحاجة لهذه المعلومات او المتغيرات فانه يتم مسحها
٤ – الجدير بالذكر ان المتغيرات انواع وليست نوع واحد فهناك المتغير الحرفى والمتغير الرقمى وغيرها.

واهم انواع المتغيرات هى :
المتغير الحرفى :
وعاء للتخزين يمكنه سعة اى نوع من البيانات (حروف – ارقام – رموز)
المتغير الرقمى :
وعاء للتخزين لا يستقبل سوى الارقام والعلامات العشرية وبعض العلامات التى تفيد معنى للرقم

المتغير التاريخى :
وعاء لتخزين التاريخ (اليوم والشهر والسنة)
المتغير المنطقى :
وعاء سعة حرف واحد لتخزين قيم من قيمتين (نعم او لا)
المتغير الشامل :
وهى شبيه بالمتغير الحرفى الا ان سعة تخزين البيانات فيه اكبر

كيفية التعامل مع المتغيرات :
ينبغى معرفة كيفية
١ – تحديد العمر الزمنى للمتغير
٢ – تحديد اسم المتغير
٣ – تحديد نوع التغير
٤ – الوصول الى البيانات داخل المتغير
٥ – تغير قيمة المتغير

اولا : تحديد العمر الزمنى للمتغير
هذا مفهوم بسيط ولكن لا يمكن عرضه الان لانه يعتمد على مفهوم اخر وهو نمط البرمجة (البرمجة الهيكلية او برمجة الكائنات – الخادم الممتاز) لذلك سوف نناقش ذلك الامر بعد صفحات.

ثانيا : تحديد اسم المتغير ونوعه وقيمته
يتم ذلك ببساطة بمجرد كتابة الاسم يليه علامة = ثم القيمة التى يحملها المتغير
فمثلا للاعلان عن متغير اسمه MyName يحمل القيمة "Mahmoud" يكون كالتالى
MyName = "Mahmoud"
ونلاحظ وضع الاسم Mahmoud بين علامتى تنصيص وذلك يعنى ان المتغير نوعه حرفى
ولاعلان المتغير الرقمى يكون بنفس الطريقة الا انه لا نستعمل علامتى التنصيص
MyNum1 = 3
وهنا وضعنا القيمة الرقمية 3 فى المتغير MyNum1
وبالنسبة للمتغير المنطقى فان المثال التالى يبين كيفية استخدامه حيث يعلن متغيرين منطقيين احدهما يحتوى على القيمة True والاخر يحتوى على القيمة False
MyTrue = .T.
MyFalse = .F.
ونلاحظ للتعبير عن القيمة True فى فيجوال فوكس برو نكتب الحرف T قبله نقطة وبعده نقطة وللتعبير عن القيمة False نكتب الحرف F قبله نقطة وبعده نقطة هكذا .F.

س : ما الفرق الجوهرى بين المتغيرات الحرفية والرقمية طالما ان كل منهما يمكنه تخزين البيانات الرقمية ؟
جـ : نعم ان كل منهما يمكنه تخزين البيانات الرقمية لكن اجراء العمليات الحسابية يقتصر فقط على المتغيرات الرقمية ولذلك اذا كنت تود تخزين بيانات رقمية لن تقوم باجراء عمليات حسابية عليها مثل رقم الهاتف فانه يمكنك تخزينها فى متغير حرفى وليست هناك حاجة لتخزينها فى متغير رقمى.

والان سوف ناخذ مثال يوضح مفهوم التعامل مع المتغيرات ويتعرض ايضا للمتغيرات التاريخية وبعض الوظائف الجديدة فى الفيجوال فوكس برو للتعرف على الموضوع بضورة جيدة.

```
Name = "Mahmoud"
Job    = "Researcher"
Age   = "1119"
Telephone = "01010101"
Years = 10
Money_Per_Year = 100000
Total = years * money_per_year
Mydate = Date()
Mytime = Time()
? "Name := " + name
? "Job := " + job
? "Age:= " + age
? "Telephone :=" + telephone
? "Total := " + STR(total)
? "Date:= " + DTOC(myDate)
? "Time :=" + mytime
```

وتنفيذ البرنامج يعطى النتيجة التالية :

```
Name := Mahmoud
Job := Researcher
Age:= 1119
Telephone :=01010101
Total :=    1000000

Date:= 05/26/06
Time :=12:29:42
```

Command

```
Name = "Mahmoud"
Job     = "Researcher"
Age    = "1119"
Telephone = "01010101"
Years = 10
Money_Per_Year = 100000
Total = years * money_per_year
Mydate = Date()
Mytime = Time()
? "Name := " + name
? "Job := " + job
? "Age:= " + age
? "Telephone :=" + telephone
? "Total := " + STR(total)
? "Data:= " + DTOC(myDate)
? "Time :=" + mytime
```

شكل ١٢ : التحويل بين المتغيرات

وسوف نستعرض الان المفاهيم الجديدة فى هذا المثال:

١ – لغة فيجوال فوكس برو ليست Case Sensitive اى ان كتابة اسم المتغير بحروف كبيرة او صغيرة لا يوثر
٢ – يمكن اجراء العمليات الحسابية على المتغيرات الرقمية كما فى حساب قيمة المتغير total والعلامات الاساسية هى + للجمع و – للطرح و * للضرب و / للقسمة
٣ – الدالة ()Date تعطى تاريخ اليوم وقد تم تسجيله فى المتغير mydate
٤ – الدالة ()Time تعطى الوقت فى صورة متغير حرفى
٥ - الدالة ()STR تحول المتغير الرقمى الى متغير حرفى وقد حدث ذلك عند تحويل المتغير TOTAL قبل عرضه
٦ – الدالة ()DTOC تحول المتغير التاريخى الى متغير حرفى كما تم عند تحويل المتغير Mydate قبل عرضه

والان سوف ناخذ مثال اخر يعرض مفاهيم جديدة للتعامل مع المتغيرات.

```
mystr = "Mahmoud"
? mystr
? UPPER(MYSTR)
? LOWER(MYSTR)
? LEFT(MYSTR,3)
? RIGHT(MYSTR,3)
? SUBSTR(MYSTR,3,2)
? LEN(MYSTR)
```

```
Mahmoud
MAHMOUD
mahmoud
Mah
oud
hm
        7
```

Command

```
mystr = "Mahmoud"
? mystr
? UPPER(MYSTR)
? LOWER(MYSTR)
? LEFT(MYSTR,3)
? RIGHT(MYSTR,3)
? SUBSTR(MYSTR,3,2)
? LEN(MYSTR)
```

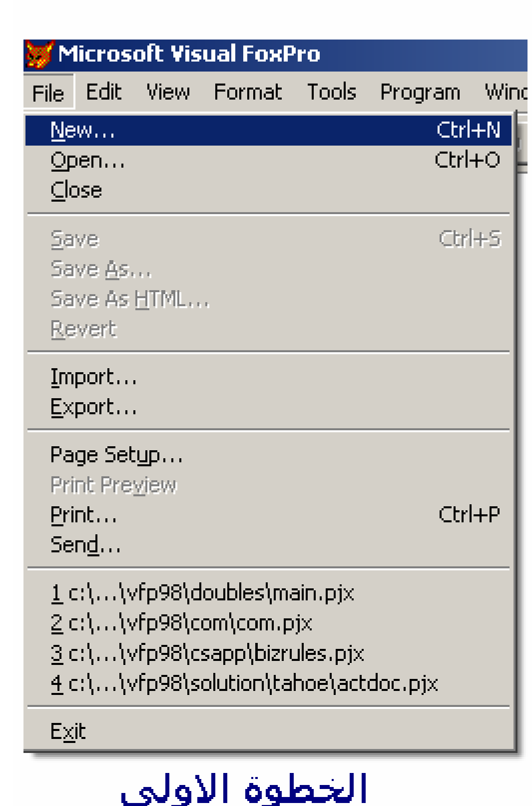
شكل ١٣ : تحليل العبارات الحرفية

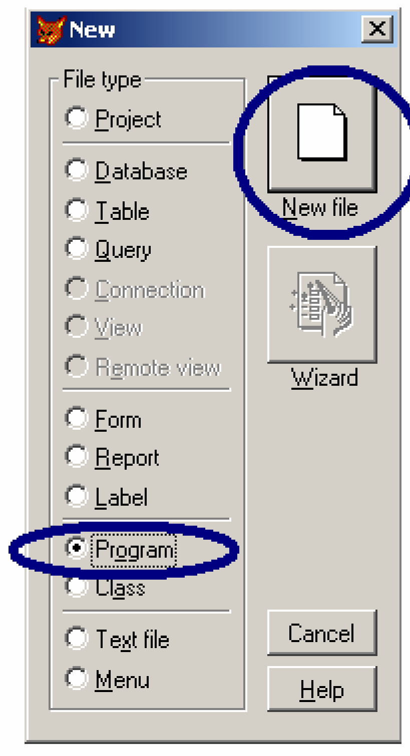
فى هذا المثال تم اعلان متغير حرفى يسمى mystr واسناد القيمة "Mahmoud" اليه وتم التعرض لستة دوال\وظائف جديدة
١ – الدالة UPPER : تقوم بتحويل العبارة الحرفية الى الحروف الكبيرة
٢ – الدالة LOWER : تقوم بتحويل العبارة الحرفية الى الحروف الصغيرة
٣ – الدالة LEFT : تقوم باستخراج جزء من العبارة الحرفية ابتداء من اليسار وتاخذ هذه الدالة اسم العبارة الحرفية وعدد الحروف المطلوب اخذها
٤ – الدالة RIGHT : تماما مثل الدالة LEFT الا انها تبدا باخذ الحروف من اليمين
٥ – الدالة SUBSTR : تاخذ جزء من العبارة الحرفية عن طريق تحديد ثلاثة معطيات الاول هو اسم المتغير الذى يحتوى على العبارة الحرفية والثانى نقطة بداية اخذ الحروف من اليسار والثالث عدد الحروف التى يتم اخذها
٦ – الدالة LEN : تعطى هذه العبارة طول المتغير

ملفات الاجراء (Procedures) :

ملفات الاجراء هى الملفات التى يتم فيها تخزين تعليمات البرنامج وتاخذ هذه الملفات الامتداد PRG.
وبذلك يمكننا الاحتفاظ بالامثلة التى قمنا بعرضها لتنفيذها فى اى وقت وكذلك تعديلها وتطويرها
وهناك عدة طرق مختلفة لانشاء ملفات الاجراء

الطريقة الاولى : كما فى شكل ١٤





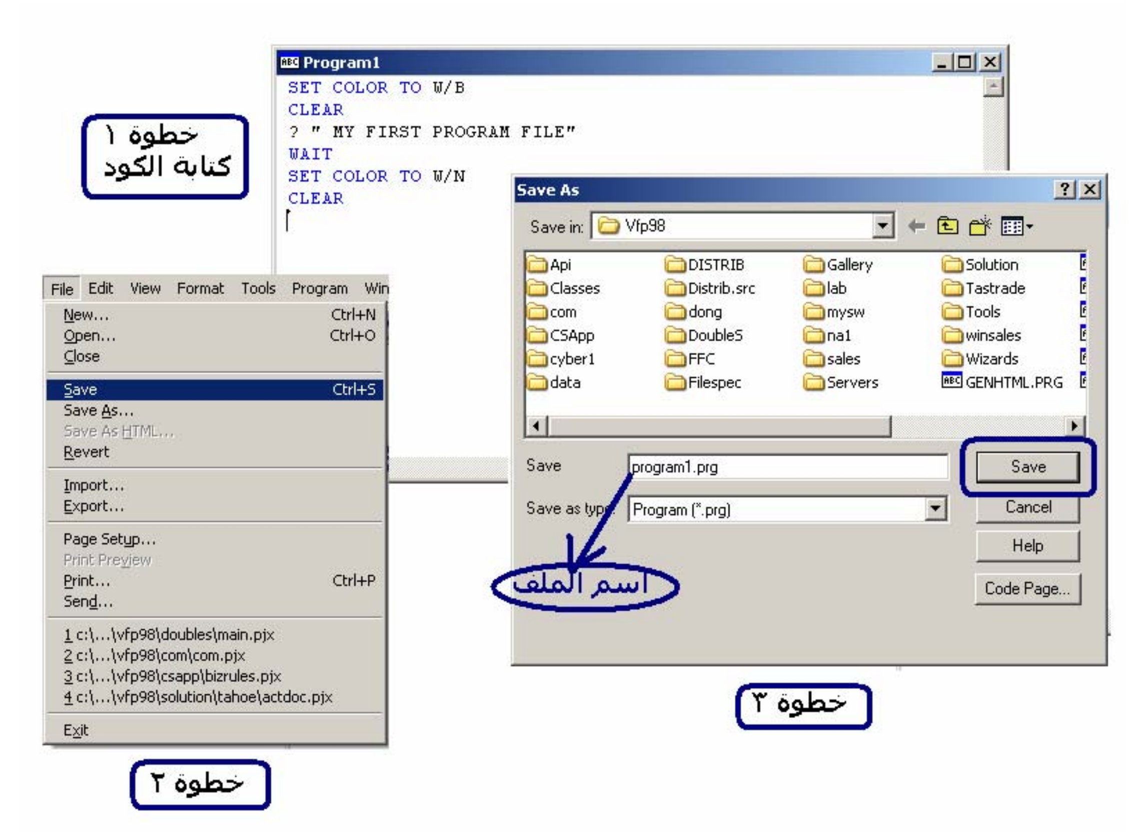
شكل ١٤ : انشاء ملف برنامج \ اجراء جديد

ونلاحظ انه يمكن اختصار الخطوة الاولى من خلال الضغط على Ctrl + N وعلامة + تعنى انه يتم الضغط على مفتاح Ctrl والاستمرار فى الضغط حتى يتم الضغط على مفتاح N
او من خلال شريط الادوات Tool Bar يتم اختيار الزر New

الطريقة الثانية : من خلال نافذة الاوامر command window
وذلك عن طريق الامر MODIFY COMMAND

ملحوظة :

تقوم فيجوال فوكس برو عند تنفيذ اى امر بوضع الامر المقابل له تلقائيا داخل نافذة الاوامر لذلك اذا قمت بانشاء ملف الاجراء من خلال الطريقة الاولى فانك سوف تجد ان الامر MODIFY COMMAND قد تم كتابته بصورة تلقائية داخل نافذة الاوامر.



شكل ١٥ : كتابة الكود داخل ملف البرنامج او الاجراء وحفظ الملف

تعدد ملفات الاجراء داخل البرنامج :

من المؤكد انه عند كتابة البرامج تظهر الحاجة لتقسم البرنامج الى مجموعة من الملفات يقوم كل جزء بوظيفة محددة وحيث ان لغة فيجوال فوكس برو من اللغات النادرة تجت وندوز التى تتيح عمل تطبيقات كاملة بالكود بدون الحاجة الى برامج التصميم (مثل مصمم النماذج – القوائم – التقارير) فانه ينبغى فهم كيفية هيكلة البرنامج من خلال الكود وخاصة ان تلك اللغة الرائعة تدعم بصورة كاملة كل من البرمجة الهيكلية ونمط برمجة الكائنات مما يعنى توفر مصادر قوة هائلة عند كتابة الكود.

الامر DO :

يقوم هذا الامر بمناداة ملف Procedure(.PRG) فرعى وتنفيذه وعند الانتهاء من التنفيذ يعود التحكم للبرنامج الرئيسى لمتابعة تنفيذ الاوامر التى تلى ذلك الامر
مثال : اذا كان لدينا برنامج رئيسى Main.prg ونود مناداة برنامج فرعى sub.prg يكون الامر كالتالى

```
Do sub
```

ويمكن كتابة الامر على الصورة do sub.prg او الامر do يليه المسار الذى يوجد فيه الملف داخل القرص الصلب

الامر Return :

يقوم هذا الامر بانهاء التنفيذ داخل الملف Procedure فاذا كان هذا الملف هو الملف الرئيسى فانه يتم بهذا الامر انهاء البرنامج واذا كان الملف ملف فرعى فانه يتم العودة للملف الذى قام بمناداته لمتابعة تنفيذ العمليات

ملحوظة:

يمكن ان ينادى الملف الفرعى ملف فرعى اخر وهكذا – اى يمكن نقل التحكم اكثر من مرة من برنامج (ملف اجراء) لاخر من خلال الامر Do والعودة خطوة للخلف من خلال الامر Return

الامر Procedure :

يقوم هذا الامر باعلان اجراء صغير داخل الاجراء الكبير (PRG.) وهذا الاجراء يمكن مناداته كما يتم مناداة اى اجراء اخر باستخدام الامر DO.
والجدير بالذكر ان الاجراء الكبير (PRG.) يمكن ان يحتوى على مجموعة من الاجراءات الصغيرة بجانب مجموعة من الوظائف لكن الاجراء الصغير لا يحتوى على اى اجراءات داخلية وكذلك لا يحتوى على وظائف داخلية
ينتهى اعلان الاجراء بالامر Return الا انه لا يرجع قيمة كما فى حالة الدالة او الوظيفة Function
الفرق بين الاجراء والدالة من حيث ارجاع القيمة هو فرق فقط من الناحية الاكاديمية وليس من الناحية العملية بمعنى يمكن عمل اجراء ويرجع قيمة ويمكن عمل دالة ولا ترجع قيمة

```
PROC <proc name>
.........code
RETURN [VALUE]
```

الامر Function :

يقوم هذا الامر باعلان دالة جديدة يمكن استخدامها مباشرة كاحد دوال لغة فيجوال فوكس برو الاساسية ويشترط ان يكون اسم الدالة الجديدة غير موجود من قبل.

```
FUNC <FUNCNAME>([PARAMETERS.....])
.........code
RETURN [VALUE]
```

ملحوظة:

يمكن ان نقوم باختصار اوامر لغة فيجوال فوكس برو من خلال كتابة الحروف الاربعة الاولى من الامر فعلى سبيل المثال الامر procedure يمكن ان سكتب proc والامر function يمكن ان يكتب func

مجال المتغيرات والدوال scope of variables & functions :

يمكن اثناء تعريف المتغيرات او الدوال تحديد مجال العمل لهذه المتغيرات او الدوال والمقصود بالمجال هو الانتشار للمتغير او الدالة داخل البرنامج ولكى نفهم ذلك ينبغى التعرض لمجموعة من الاوامر الجديدة او التعليمات وهى Local,Private,Public & Static وهى اربعة اوامر تحدد مجال انتشار المتغير والامر Static يستخدم لتحديد مجال انتشار الدوال

استخدام الامر Static قبل الدوال المعرفة داخل اجراء كبير (ملف Prg.) يعنى ان هذه الدالة لن يتم معرفتها الا داخل هذا الملف او الاجراء واذا كان هناك اكثر من ملف Prg. داخل البرنامج فان هذه الملفات الاخرى لن تشعر بوجود هذه الدالة
س : ما فائدة تعريف دالة بالامر static ؟
جـ : يتيح لك امكانية تعريف الدالة بنفس الاسم اكثر من مر ة بحيث كل مرة داخل ملف prg. مستقل

استخدام الامر static قبل اعلان المتغير اى عند تعريفه اول مرة يعنى ان هذا المتغير لن تتغير قيمته وانه سيظل متاح امام كافة الاجراءات سواء كانت ملفات prg. او اجراءات فرعية او دوال داخل البرنامج ان تعرف قيمة هذا المتغير ولكن لن يكن متاح امامها ان تعدل فى قيمته

استخدام الامر local امام المتغير يعنى انه داخل الدالة التى تم تعريفه داخلها وغير متاح لاى دوال فرعية (الدوال التى يتم مناداتها من داخل الدالة التى تعمل) ان تتعامل معه
وكذلك فانه بمجرد انتهاء تنفيذ الدالة التى اعلنت هذا المتغير فانه يتم حذف المتغير ويصبح غير متاح

استخدام الامر private يعنى ان المتغير متاح داخل الدالة و للدوال الفرعية وغير مسموح لبقية اجزاء البرنامج الاخرى التعامل معه
وكذلك فانه بمجرد انتهاء تنفيذ الدالة التى اعلنت هذا المتغير فانه يتم حذف المتغير ويصبح غير متاح

استخدام الامر public يعنى ان المتغير متاح داخل كامل اجراء البرنامج ويمكن تعديل قيمة المتغير من اى جزء فى البرنامج .

```
PUBLIC/STATIC/LOCAL/PRIVATE <VARNAME> [= <VALUE>]
```

ملحوظة:

اوامر المجال scope الخاصة بالمتغيرات يمكن استخدامها قبل اسم المتغير مع اسناد قيمة له فى نفس السطر او مع اسم المتغير بدون اسناد اى قيمة له
العلامتين <> تعنى ان مابداخلهما ضرورى و العلامتين [] تعنى ان مابداخلهما اختيارى والعلامة / تعنى(او) اى يتم اختيار واحد مما بين الخيارات التى تفصلها العلامة /

تركيبات التحكم Control Structure:

احدى اروع سمات البرمجة التى من خلالها يمكن الحصول على النتائج المطلوبة من البرامج التى يتم كتابتها واهم هذه التركيبات if-else-endif و for-endfor و do while-enddo و case-endcase

تتيح جملة if تنفيذ تعليمات معينة اذا تحقق شرط محدد واذا لم يتحدد يتم تنفيذ التعليمات التى تلى جملة else وهى اختيارية
جملة for تستخدم لتكرار تنفيذ العمليات عدد محدد من المرات
Do while تستخدم لتنفيذ عدد من العمليات مادام ان هناك شرط ما متحقق
جملة case تستخدم فى الاحتمالات وغالبا الاحتمالات المبنية على متغيرات رقمية

* التفرع المشروط

```
If <condition - expression>
 ........code
else
 ........code
endif
```

* التكرار المشروط من خلال المتغيرات الرقمية

```
for <Nvariable> = <Nvalue> to <Nvalue> step <Nvalue>
.....code
endfor
```

* التكرار المشروط

```
do while <condition - expression >
.......code
enddo
```

* التفرع المشروط من خلال المتغيرات

```
do case
    case <variable> = value
     .......code
    case <variable> = value2
    ........code
endcase
```

التعبيرات Expression :

تستخدم التعبيرات للحصول على نتيجة معينة من عدة متغيرات
واذا وجد التعبير مع جملة if او while فانه يسمى condition كانه سؤال للحاسب او للغة البرمجة التى تعطى النتيجة .T. اذا تحقق الشرط او .F. اذا لم يتحقق
البعض يظن انه ليس هناك نتيجة اخرى لكن فى الواقع هناك احتمال ثالث وهو ان تحصل على ERROR اى رسالة خطا وذلك اذا كان التعبير غير سليم.

الاشىء او العدم :

المتغير عند اعلانه اول مرة بدون اسناد قيمة له فان لغة البرمجة فيجوال فوكس برو لن تعرف نوعه لانه تحدد نوع المتغير تبعا للقيمة التى يحملها ولهذا سوف تضع فى المتغير قيمة منطقية وهذه الفيمة هى .F.
ولكن اثناء العمل مع المتغيرات واذا اردنا جعل قيمة المتغير لاشى فاننا نسند له القيمة NULL والتى تعنى العدم

العمل المستمر :

عند تشغيل البرنامج فانه يتم تنفيذ تعليماته تعليمة تلو الاخر حتى يتم الانتهاء من التنفيذ ومن ثم الخروج من البرنامج ولضمان استمرار عمل البرنامج فانه تم كتابة التعليمات بين while و enddo ويكون الشرط هو .T. وهو ابسط شرط ويعنى ان الشرط دائما متحقق

التحكم فى عمل الحلقة :

الامر EXIT : اذا استخدم داخل الحلقة فانه ينهى العمل داخل الحلقة سواء كانت DO WHILE او FOR وينقل التنفيذ لما بعد الحلقة
الامر LOOP : اذا استخدم داخل الحلقة فانه ينقل التنفيذ للمرحة الاولى مرحلة اختبار الشرط ويهمل التعليمات التى تلى ذلك الامر

والان سوف ننتقل لمثال جيد وهو عبارة عن برنامج مكتوب داخل ملف PRG. وليكن Main.Prg
والكود او التعليمات الموجودة بداخله هى :

```
do while .T.
        set color to bg+/b,gr+/rb
        clear
        @1,2 say "يفصل البحر الاحمر بين دولتى مصر و السعودية " font "",20
        @7,12 prompt "            نعم العبارة صحيحة          " font "",30
        @10,12 prompt "           لا العبارة خطا      "
        @13,12 prompt "              خروج           "
        menu to mymenu
        do case
         case mymenu = 1
                                myop1()
          case mymenu = 2
                               do myproc2
          case mymenu = 3
                                 exit
        endcase
enddo
set color to w/n
clear
? "مع السلامة "
return
function myop1()
  set color to w/r
  clear
  ? "احسنت الاختيار"
  wait
return
proc myproc2
  set color to w/n
  clear
  ? "الاجابة خطا"
  wait
return
```

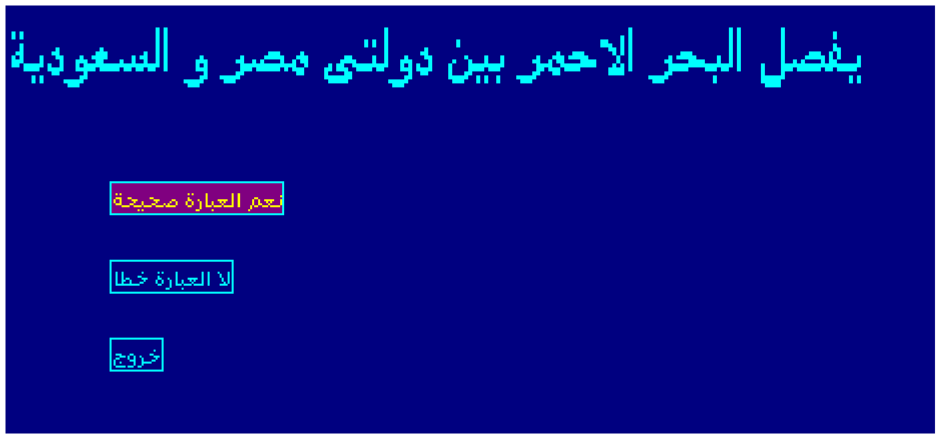
وهذا البرنامج هو اول تطبيق بسيط على ما تم شرحة الا اننا تعرضنا للامر say والذى يعرض نص محدد على الشاشة فى مكان محدد (السطر ثم العمود) فاول جملة say تعرض سؤال فى السطر الاول والعمود الثانى
المر prompt مشابه لامر say الا انه يستخدم لعمل خيارات القائمة التى تعمل بالشريط المضىء

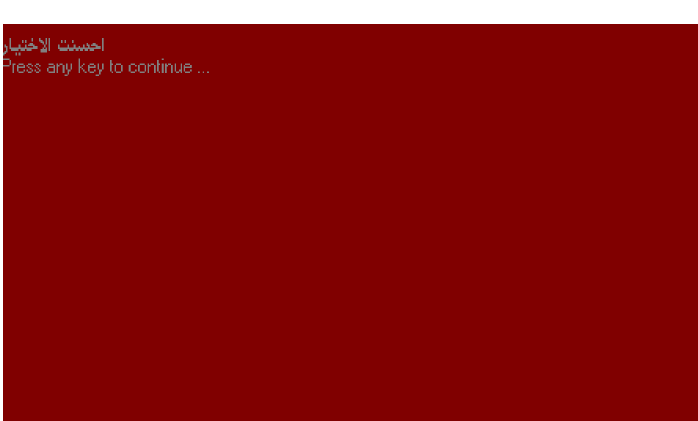
الامر menu to يستخدم لتحدد المتغير الرفمى الذى سوف يحمل نتيجة الاختيار الذى قام به المستخدم
ونلاحظ انه لاستجابة التنفيذ للاختيار الذى قام به المستخدم قمنا مرة بنداء اجراء فرعى من خلال الامر DO ومرة اخرى قمنا بمناداة دالة فرعية قمنا بتعريفها

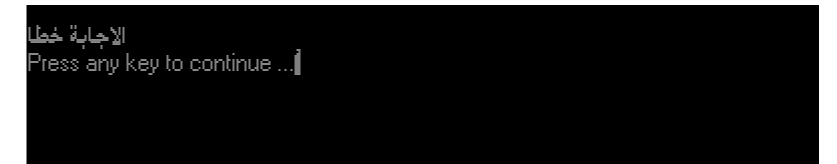
شكل ١٦ يوضح كيفية تنفيذ البرنامج الذى تم كتابته فى اى وقت
شكل ٧٦ يوضح نتيجة عمل البرنامج



شكل ١٦ : تنفيذ برنامج مكتوب فى اى وقت







شكل ١٧ : نتيجة تنفيذ البرنامج

والان سوف ناخذ مثال على الامر Private

PRIVATE Command Example

```
*** Program example demonstrating PRIVATE ***
val1 = 10
val2 = 15

DO down
? val1, val2  && Displays 10, 100

PROCEDURE down
PRIVATE val1
val1 = 50
val2 = 100
? '   Val1   Val2'
? val1, val2  && Displays 50, 100
RETURN
```

فى هذا المثال تم تعريف متغيرين فى البرنامج الرئيسى (.prg) وهذين المتغيرين هما val1,val2 وتم اسناد قيم اليهما ثم تم مناداة اجراء فرعى باسم down وهذا الاجراء عرف المتغير val1 على انه private لذا فانه لن يتداخل مع المتغير val1 الذى تم تعريفه فى الاجراء الرئيسى ويحمل نفس الاسم بينما الاجراء الفرعى لم يعرف المتغير val2 ولهذا فان المتغير val2 فى الاجراء الفرعى هو نفسه المتغير val2 الموجود فى الاجراء الرئيسى .

الحصول على المساعدة :

العمل فى مجال البرمجة فى البداية يتطلب الكثير من المساعدة من خلال
١ – استشارة صديق (والذى لا تجده متفرغا لك دائما)
٢ – الدورات التدريبية (والتى قد تكون مكلفة وغير كافية)
٣ – الكتب المطبوعة (والتى يندر الحصول على كتاب جيد ومتكامل)
٤ – الكتب الالكترونية (والتى تكون اقل جودة بكثير من الكتب المطبوعة)
٥ – الانترنت (وهو مصدر المقالات والكتب الالكترونية)
٦ – شاشات المساعدة الخاصة باللغة **وهى الصديق المتفرغ لك تماما والدورة المجانية الكافية لاشباع احتياجاتك .**

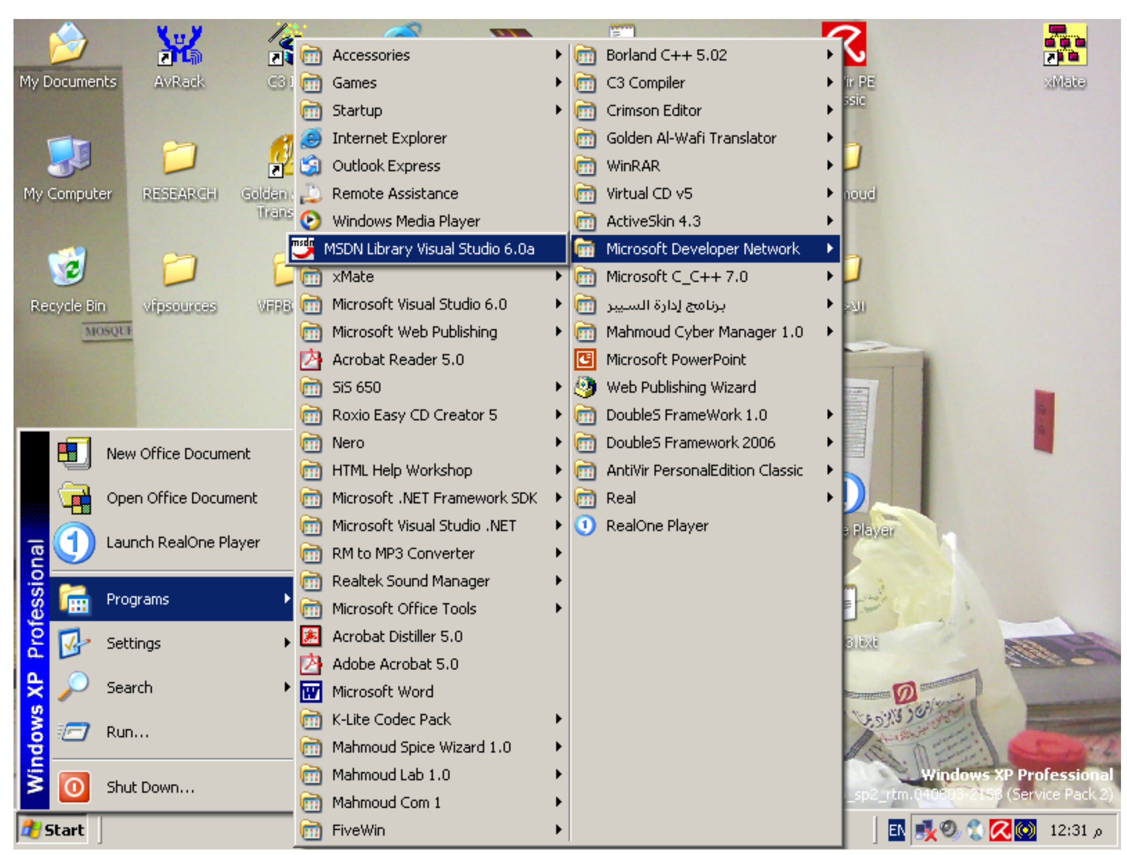
ولهذا فان ايجاد الحصول على المعلومات المطلوبة امر فى غاية الاهمية

ومن هنا سوف ناخذ جولة فى MSDN Library Visual Studio 6

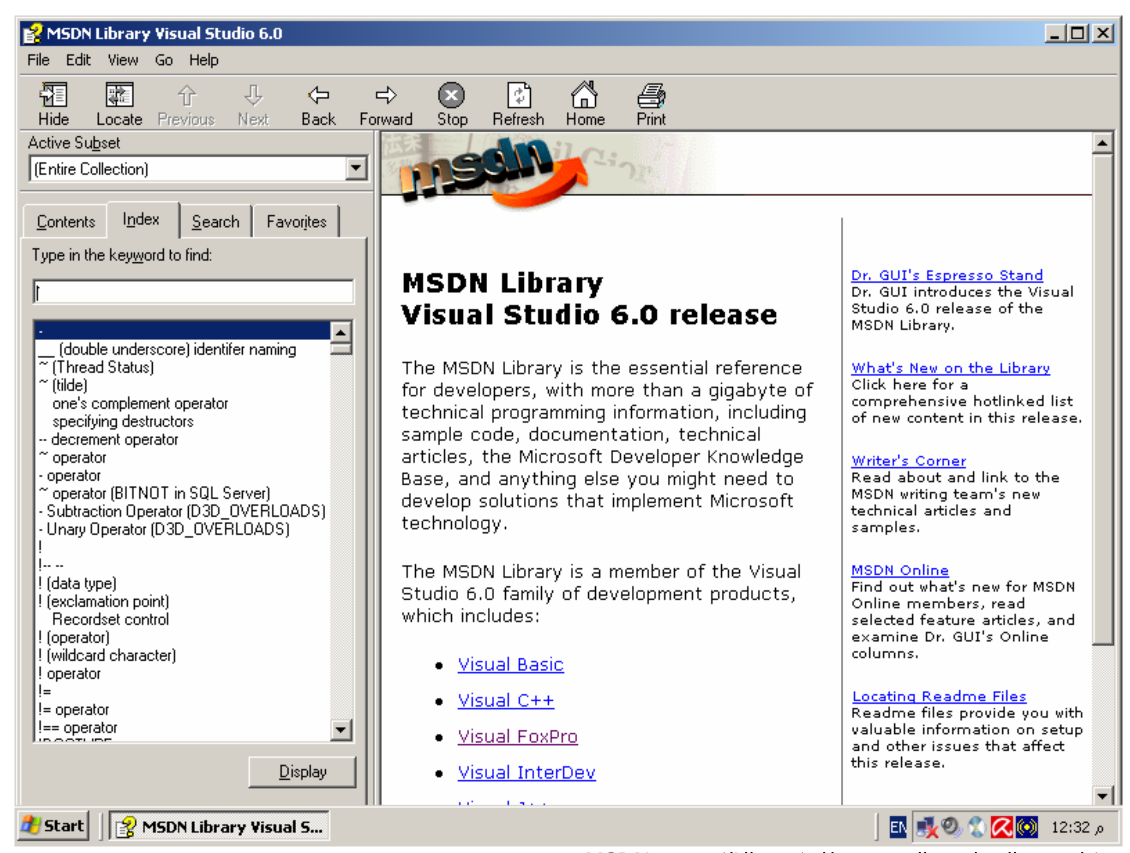
شكل ١٨ يوضح كيفية التشغيل وشكل ١٩ يوضح الشكل الرئيسى .

والمكتبة الالكترونية تشمل لغات محيط التطوير وليس Visual FoxPro فقط ومن خلال المكتبة الالكترونبة يمكنك التعلم خطوة بخطوة وسوف تجد الكثير من المعلومات التى لا يسع هذا الكتيب الصغير ان يذكر ١ % منها ومن خلال تلك المكتبة يمكنك التعلم خطوة بخطوة والمضى نحو الاحتراف

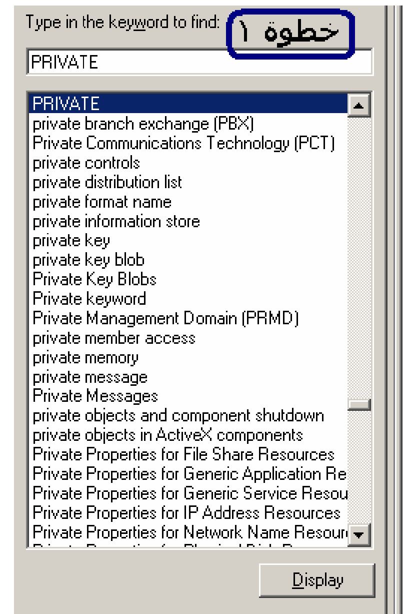
وايضا من اهم المميزات البحث عن معلومة بصورة مباشرة وسوف ناخذ مثال بالبحث عن الامر PRIVATE والذى قد تم الاشارة اليه سابقا.

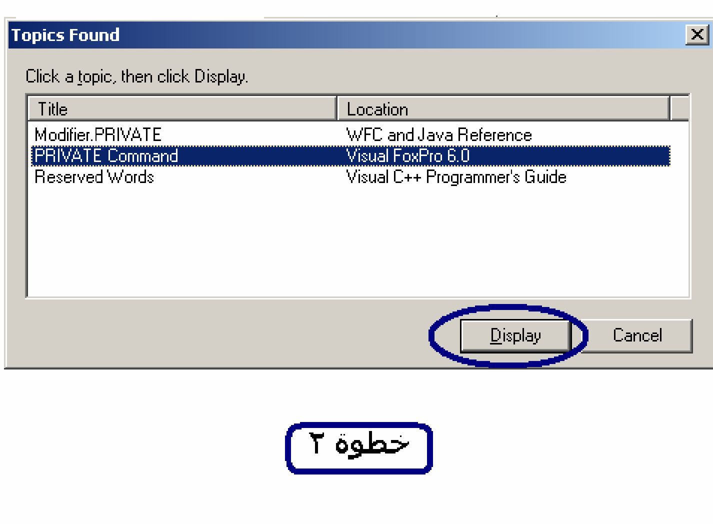


شكل ١٨ : تشغيل المكتبة الالكترونية MSDN



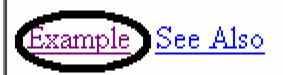
شكل ١٩: الشاشة الرئيسية للمكتبة الالكترونية MSDN





شكل٢٠: البحث عن امر PRIVATE

PRIVATE Command



Hides specified variables or arrays that were defined in a calling program from the current program.

Syntax

PRIVATE *VarList*

-or-

PRIVATE ALL
[LIKE *Skeleton* | EXCEPT *Skeleton*]

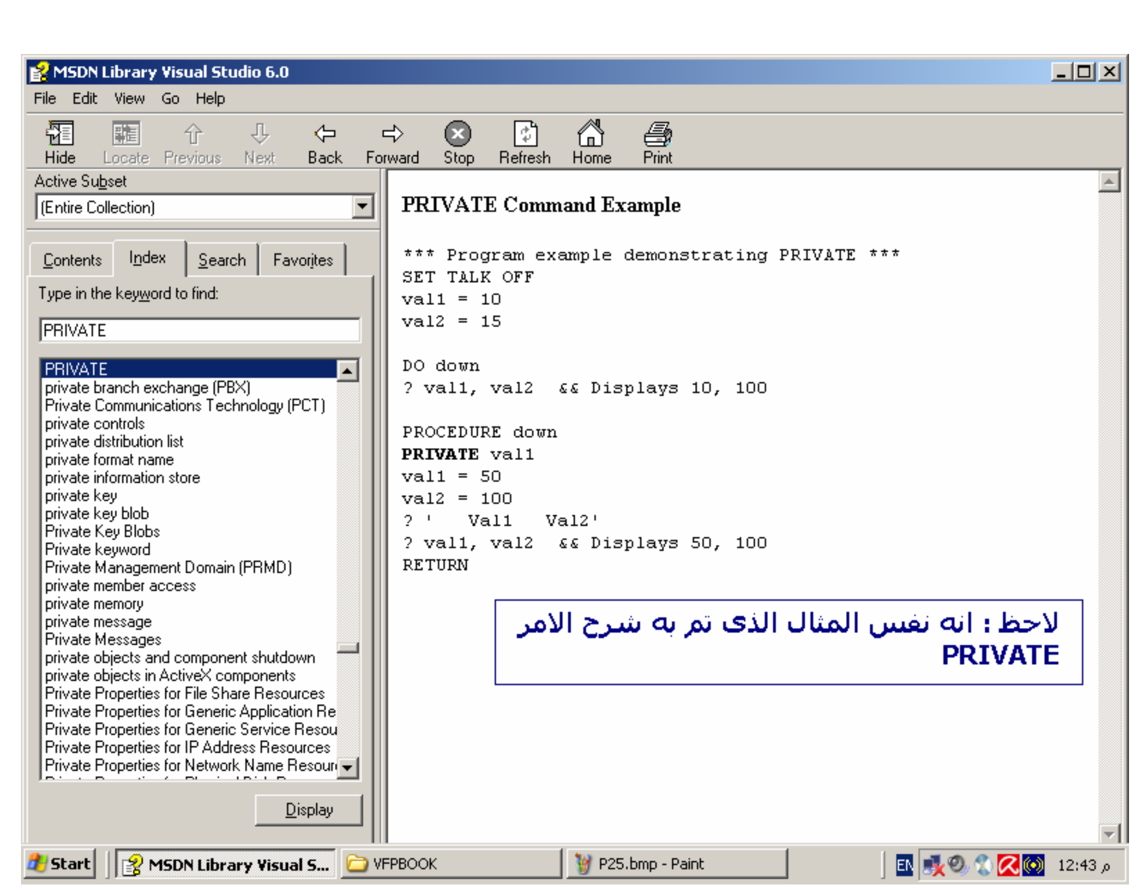
Arguments

VarList

Specifies the variables or arrays to be declared private.

ALL LIKE *Skeleton*

شكل٢١: الحصول على معلومات تشرح الامر PRIVATE



شكل٢٢: الحصول على مثال على الامر PRIVATE



الباب الثا

ملفات البيانات

فى هذا الباب نود عمل جولة سريعة جول كيفية انشاء ملفات البيانات واضافة وتحديث البيانات التى بداخلها بطريقتين الطريقة الاولى من خلال محيط تطوير فيجوال فوكس برو والطريقة الثانية من خلال ملفات الاجراءات (البرامج) عن طريق كتابة التعليمات (الكود).

ملف البيانات داخل فيجوال فوكس برو هو ملف يحمل الامتداد DBF. وهى اختصار ل (DataBase File) ولهذا الملف مواصفات يقوم الشخص الذى ينشئه بتحديدها هذه المواصفات تسمى Details وهى عبارة عن اسماء الحقول Data Fields وانواعها (حرفى – رقمى – وغيرها) والسعة التخزينية للحقل

فمثلا اذا كنت تود انشاء ملف بيانات يحمل ارقام هواتف الاصدقاء فانت بحاجة على الاقل الى تسجيل الاسماء بجانبها الارقام وهنا نضع فى ملف البيانات حقلين احدهما يحمل الاسم والاخر يحمل رقم الهاتف ويكون نوع الحقل الاول حرفى لان الاسم عبارة عن حروف والاخر ايضا حرفى لان رقم الهاتف ليس رقم حسابى ولن نحتاج لاجراء العمليات الحسابية عليه ويكون سعة حق الاسم مثلا ٥٠ حرف وحقل رقم الهاتف ١٥ ولا اظن اننا نحتاج اكثر من ذلك.

ان ملف البيانات لا تقتصر مهمته فقط على حفظ البيانات لاسترجاعها فى اى وقت بل يتيح لنا البحث داخله للحصول على المعلومة المطلوبة وفى نظم قواعد البيانات التى تشتمل على العديد من ملفات البيانات فى القدرة على استخلاص المعلومات من هذه الملفات التى تعمل معا امر فى غاية الاهمية

ولما كانت عملية البحث هى عملية شائعة فى جميع تظم قواعد البيانات فان العمل على زيادة سرعة تلك العملية امر فى غاية الاهمية ومن هنا نشأ مفهوم ملفات الفهارس والتى تقوم على فهرسة بيانات الملف من خلال بيانات حق معين وتكون عملية البحث فى ملفات الفهارس اسرع بكثر من البحث فى ملف البيانات بصورة مباشرة

ولما كان الاستعلام او البحث داخل ملف البيانات قد يكون غاية فى البساطة احيانا وقد يصل لاعلى مستويات التعقيد فى الانظمة الكبيرة نشأت الحاجة لملفات تشمى ملفات الاستعلام والتى تساعد على عملية البحث

ونظرا لان ثمرة عمل انظمة قواعد البيانات تكمن فى التقارير التى يمكن استخراجها من البيانات مباشرة وبسرعة فائقة لذا فان وجود ملفات التقارير التى تقوم بتلك المهمة كان امرا طبيعيا وضروريا فى نفس الوقت لسرعة وسهولة الاداء

وحيث ان ملفات البيانات قد توجد روابط بينها لكفاءة الاداء من هنا نشاء مفهوم العلاقات بين الجدوال والتى تؤدى الى حل كثير من المشاكل وتساهم بصفة رئيسية فى امبراطورية قواعد البيانات والتى لهذا السبب تسمى قواعد بيانات علاقية.

تتميز فيجوال فوكس برو كلغة برمجة متكاملة باحتوائها على نظام ادارة قواعد بيانات متكاملة داخل محيط تطوير اللغة مما لا يجعل هناك حاجة للخروج عن محيط تطوير اللغة لانشاء ملفات البيانات والاستعلام والفهارس او التقارير حيث ان كل شىء يمكن التعامل معه مباشرة داخل اللغة كما انها تتيح برمجة البيانات والتعامل معها على مستوى عالى جدا من الكفاءة.

وفى حالة بناء الانظمة الكبيرة والتى تحتاج قواعد بيانات اكثر فاعلية وتتيح سعة تخزين عالية فان ذلك ممكن حيث يمكنك تحويل قاعدة البيانات الى SQL او Oracle ومع ذلك يظل البرنامج كما هو تقوم بتطويره بلغة فيجوال فوكس برو بنفس التعليمات.

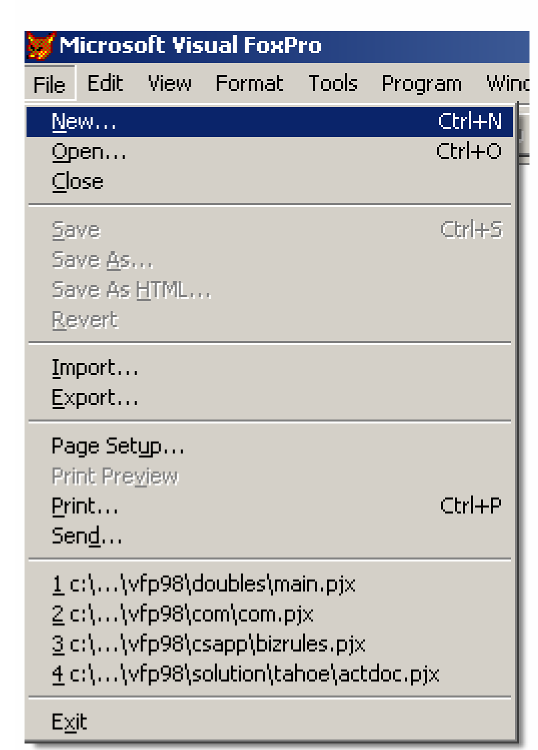
فى الواقع ان فيجوال فوكس برو يمكن اعتبارها (فيجوال بيسك + اكسس) اى لغة برمجة ونظام ادارة قواعد بيانات فى نفس الوقت

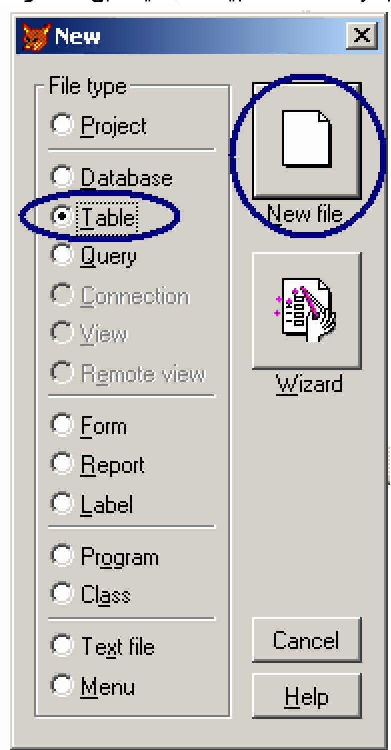
فى هذا الباب سوف يتم التركيز على ملف البيانات للقيام بالعمليات التالية :

١ – انشاء الملف ٢ – اغلاق الملف ٣ – فتح الملف
٤ – اضافة البيانات ٥ – عرض البيانات ٦ - تعديل البيانات
٧ – حذف البيانات
٨ – معرفة رقم السجل وعدد السجلات
٩ – التنقل بين السجلات
١٠- برنامج الاضافة ١١ – برنامج التعديل ١٢ – برنامج البحث
١٣– برنامج الحذف
١٤ - تعديل مواصفات الملف
١٥ – فتح اكثر من ملف بيانات فى نفس الوقت
١٦ – فتح الملفات لاكثر من مستخدم
١٧ – نسخ البيانات من ملف لاخر
١٨ – العلاقات بين الجدوال (ملفات البيانات)
١٩ – قاعدة البيانات Database والاستعلام وجمل SQL
٢٠ – التعامل مع البيانات البعيدة Remote Data

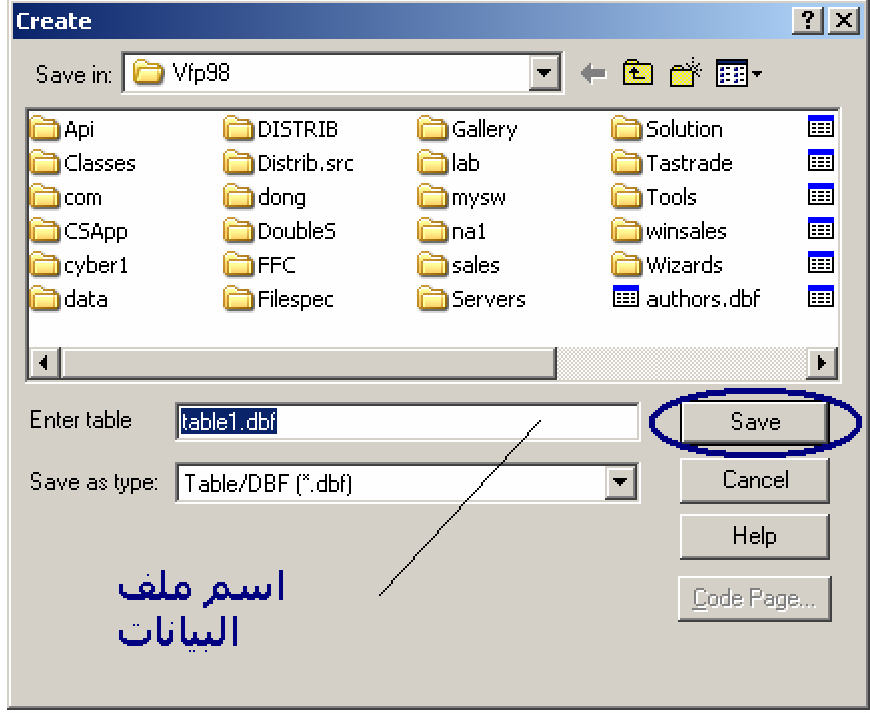
انشاء ملف البيانات (جدول) :

لكى يتم انشاء ملف بيانات جديد اتبع الخطوات التالية

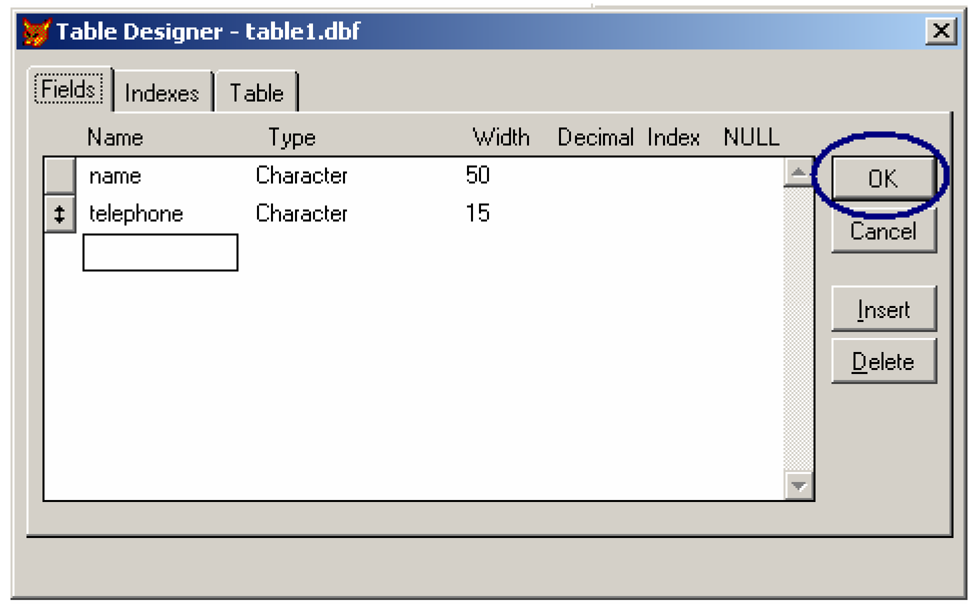




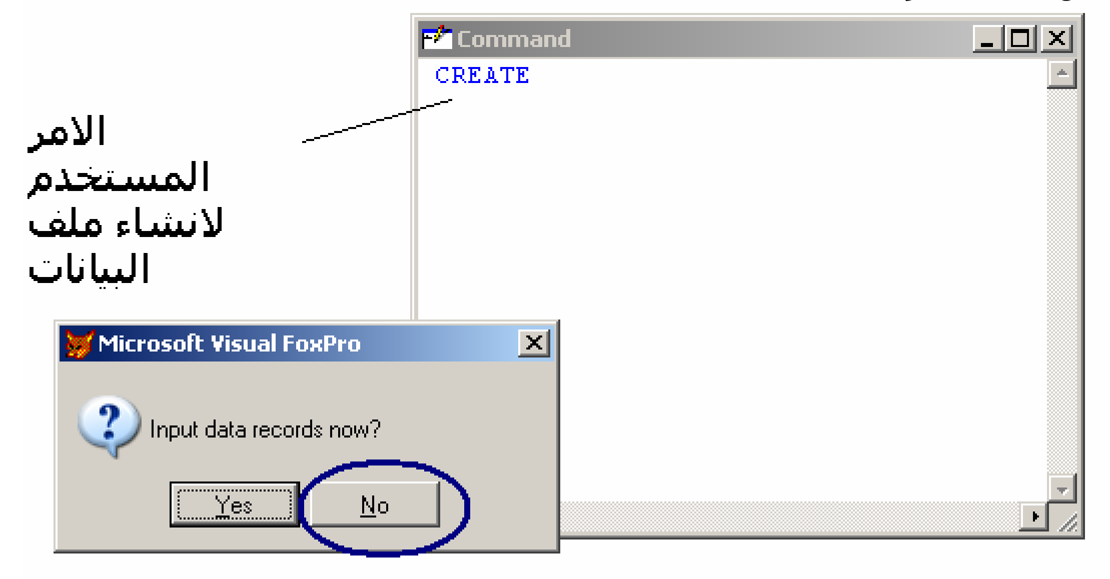
شكل٢٣ : انشاء ملف بيانات جديد (انشاء جدول).



شكل٢٤: اختيار اسم ملف البيانات



شكل ٢٥ : ادخال مواصفات ملف البيانات



شكل ٢٦ : رسالة سوال (هل تريد ادخال سجلات الان)

اغلاق ملف البيانات (جدول) :

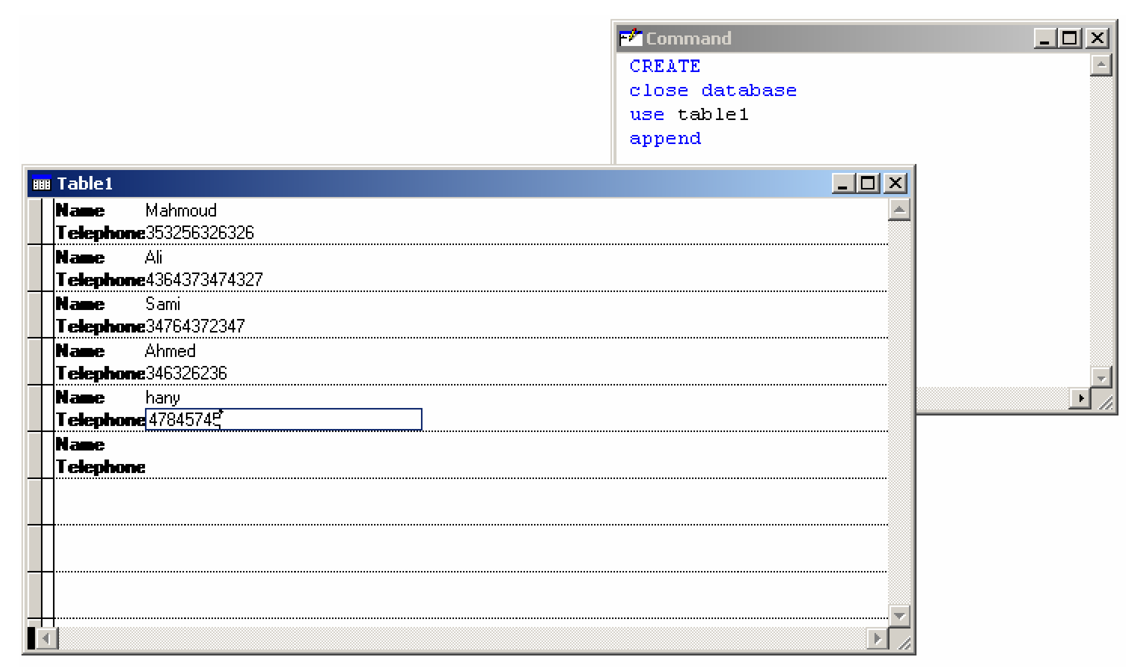
لكى يتم اغلاق ملف البيانات ندخل الامر use بدون اى معطيات

فتح ملف البيانات (جدول) :

لكى يتم فتح ملف البيانات نستخدم الامر use يليه اسم الملف
ولفتح الملف table1 الذى قمنا بانشائه يكون الامر كالتالى use table1
ويمكن فتح الملف من خلال القائمة File ثم Open واختيار الملف المطلوب

إضافة البيانات :

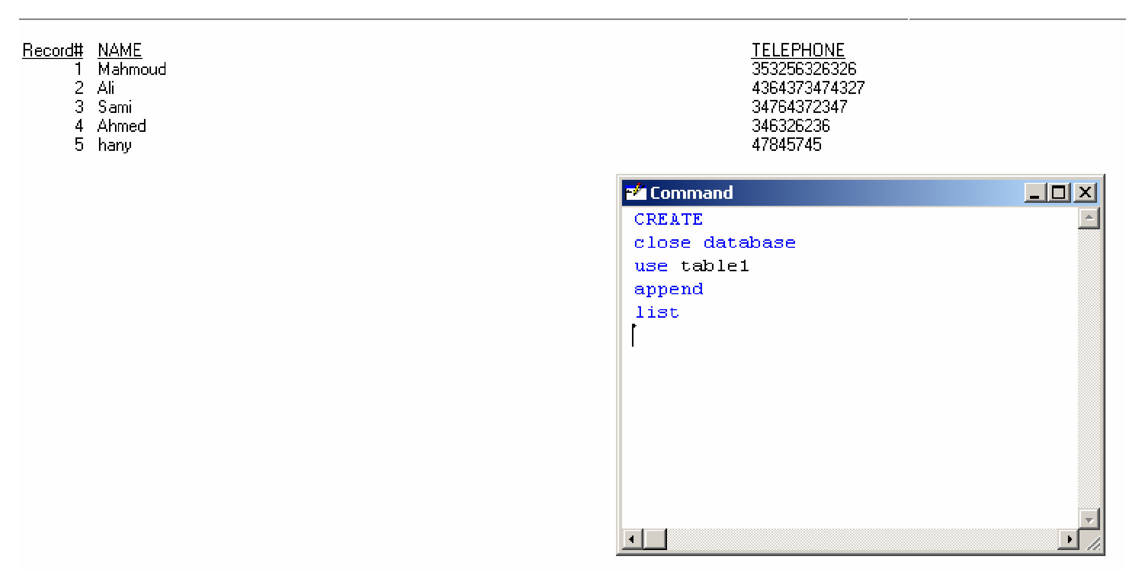
لكى يتم اضافة بيانات الى ملف البيانات المفتوح هناك طريقتين
١ – استخدام الامر append وسوف يظهر شاشة ادخال ويستخدم ذلك اثناء التجارب والعمل مع قاعدة البيانات
٢ – استخدام الامر append blank وسوف يقوم بانشاء سجل فارغ ولن يظهر شاشة لادخال البيانات حيث يتم ادخال البيانات من خلال اوامر البرمجة الخاصة باللغة وسوف نتعرض له فى برنامج الاضافة



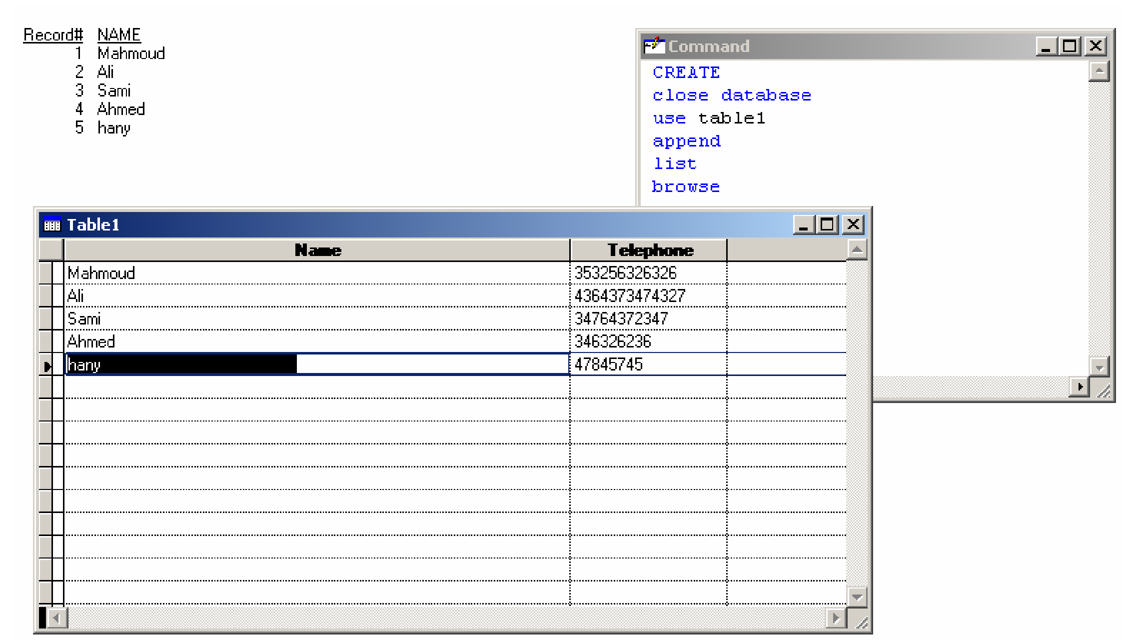
شكل ٢٧ : ادخال البيانات من خلال الامر append

عرض البيانات :

لكى يتم عرض البيانات يمكن استخدام الامر List او الامر browse



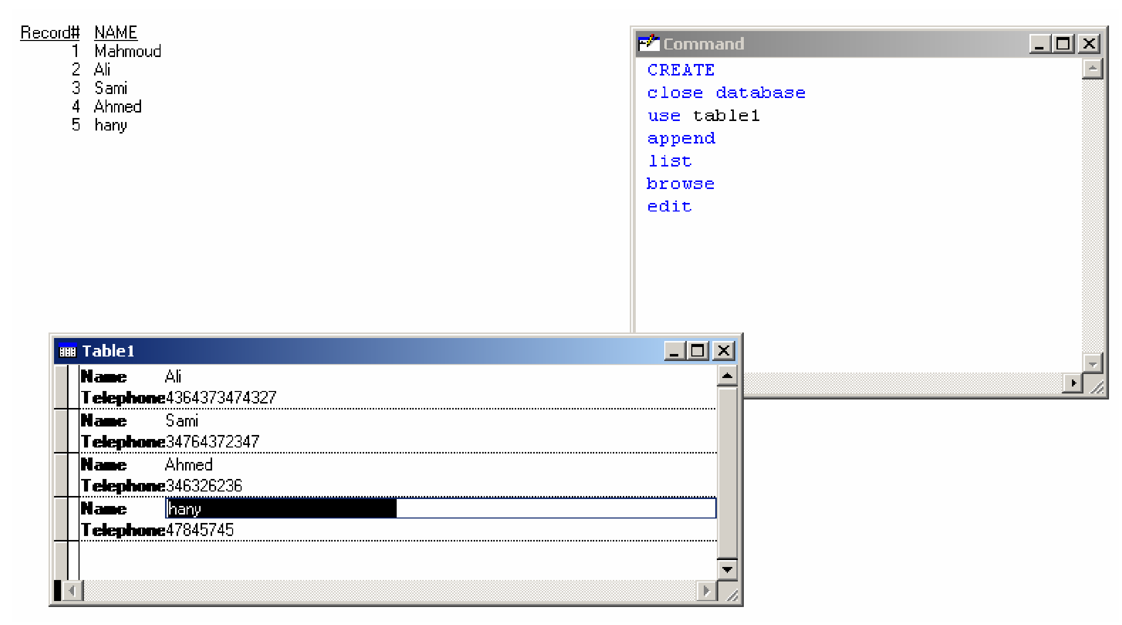
شكل ٢٨ : عرض البيانات باستخدام الامر list



شكل ٢٩ : عرض البيانات باستخدام الامر Browse

تعديل البيانات :

لكى يتم تعديل البيانات يمكن استخدام الامر Edit

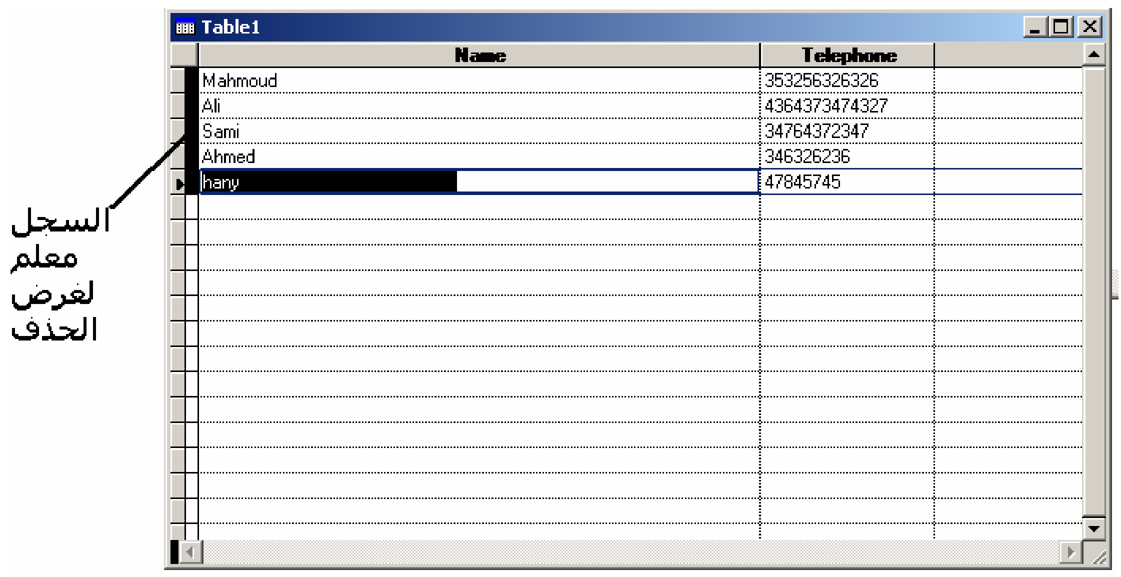


شكل ٣٠ : تعديل البيانات باستخدام الامر Edit

حذف البيانات :

لكى يتم حذف سجل يتم اولا تعليمه لغرض الحذف باستخدام الامر Delete ثم حذف السجلات المعلمة لغرض الحذف باستخدام الامر Pack وهناك مفاهيم تقف خلف ذلك سوف نتعرض لها فيما بعد ولكن ينبغى معرفة انه لحذف السجل الحالى نستخدم الامر Delete ولحذف جميع السجلات نستخدم الامر Delete All ولحذف السجلات نهائيا نستخدم الامر Pack ولاسترجاع السجل الحالى المعلم لغرض الحذف نستخدم الامر Recall ولاسترجاع جميع السجلات المعلمة لغرض الحذف نستخدم الامر Recall all
السجل المعلم لغرض الحذف تظهر امامه علامة سوداء عند استعراضه من خلال الامر Browse
للفهم استخدم الامر delete all ثم browse ثم recall all ثم browse.

يستخدم الامر set delete on لمعاملة السجلات المعلمة لغرض الحذف كانها محذوفة نهائيا ولا يتم رؤيتها ولكى نتراجع عن ذلك نستخدم الامر set delete off



شكل ٣١ : تم استخدام الامر Delete ALL ثم الامر Browse

فى شكل ٣١ قمنا بتعليم جميع السجلات لغرض الحذف باستخدام الامر Delete all ثم استعراض البيانات من خلال الامر Browse مما ادى لوجود علامة سوداء امام جميع السجلات تفيد بانها معلمة لغرض الحذف والان

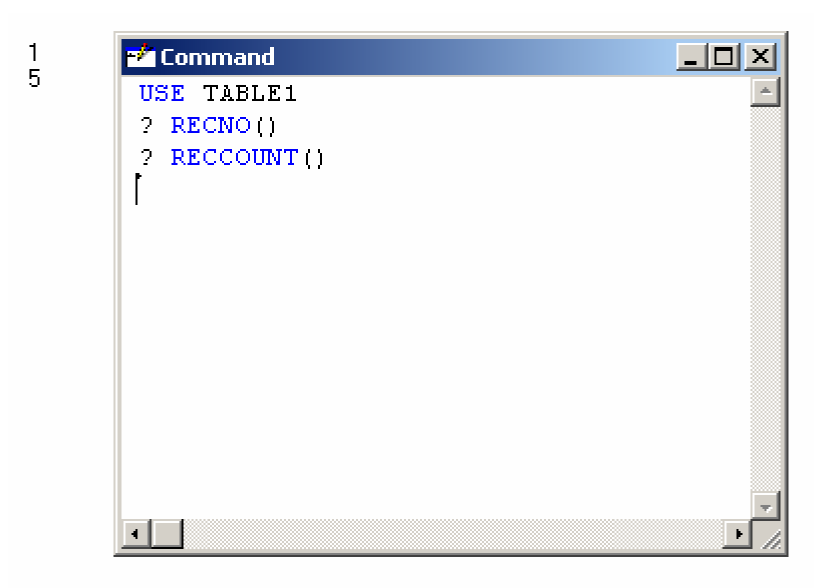
١ - يمكن استخدام الامر Recall All ثم browse لملاحظة اختفاء العلامة
٢ - او يمكنك استخدام الامر Pack لحذف السجلات ثم browse لتجد الملف فارغ

ملحوظة : لحذف جميع السجلات مباشرة يمكن استخدام الامر ZAP والذى يكون بديل للامرين Delete All ثم Pack

معرفة رقم السجل وعدد السجلات :

لكى نعرف رقم السجل الحالى والذى يقف عنده المؤشر ويمكن التعامل معاه مباشرة نستخدم الدالة RECNO()
ولكى نعرف عدد السجلات نستخدم الدالة او الوظيفة RECCOUNT()

1
5



شكل ٣٢ : معرفة رقم السجل وعدد السجلات

التنقل بين السجلات :

الامر GOTO يستخدم للذهاب لرقم سجل معين مثل GOTO 3 يذهب للسجل الثالث
الامر GOTO TOP يذهب الى اول سجل
الامر GOTO BOTTOM يذهب الى اخر سجل
الامر SKIP يقفز عدد محدد من السجلات فمثلا SKIP 1 ينقل المؤشر الى السجل التالى
والامر SKIP -1 ينقل المؤشر الى السجل السابق

```
1
5
5
1
2
1
3
```

```
Command
? RECNO()
? RECCOUNT()
GOTO BOTTOM
? RECNO()
GOTO TOP
? RECNO()
SKIP 1
? RECNO()
SKIP -1
? RECNO()
SKIP 2
? RECNO()
```

شكل ٣٢ : التنقل بين السجلات

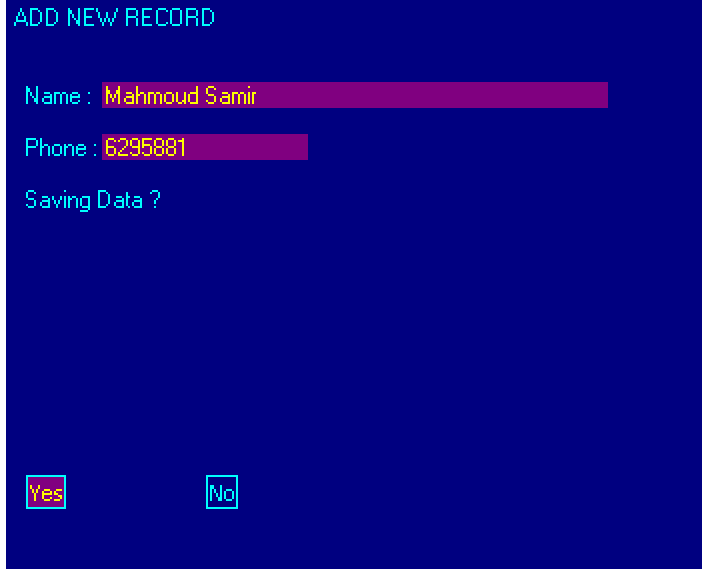
برنامج الاضافة :

تعرض شاشة ادخال للمستخدم لكى يدخل بيانات السجل واذا اراد حفظ البيانات تستخدم الامر append blank لاضافة السجل ثم تستخدم الامر replace with لنقل بيانات حقل محدد الى ملف البيانات
والبرنامج التالى عبارة عن برنامج بسيط (شاشة اضافة) ولكنها تفترض ان ملف البيانات مفتوح
لذا لا يوجد امر use فى البرنامج

- add – program

```
set color to bg+/b,gr+/rb
clear
V_name = space(50)
V_Phone = space(20)
@ 1,1 say "ADD NEW RECORD"
@4,2 say "Name : " GET v_name
@6,2 say "Phone :" GET v_phone
Read
@8,2 say "Saving Data ?"
@19,2 prompt "Yes"
@19,20 prompt "No"
menu to yn
if yn = 1
  append blank
  replace name with v_name
  replace telephone with v_phone
endif
```

والشكل التالى يوضح نتيجة تنفيذ البرنامج



شكل ٣٤ : برنامج الاضافة

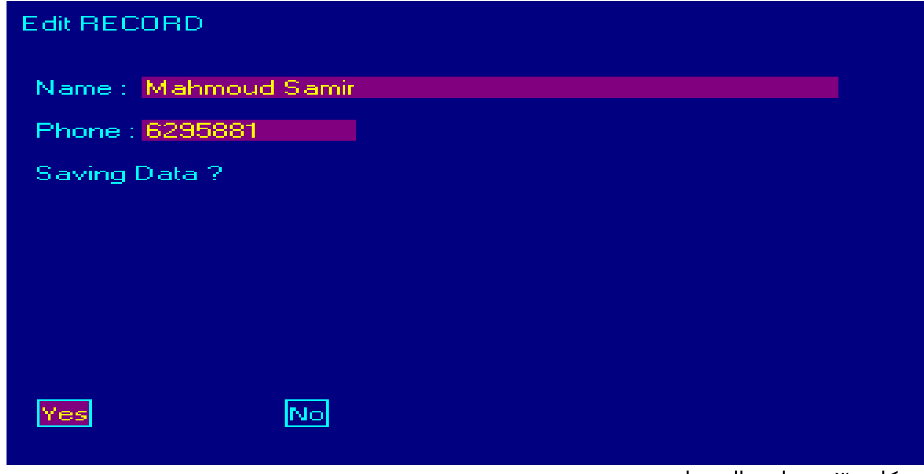
برنامج التعديل :

هو شبيه ببرنامج الاضافة الا انه لا يحتوى على الامر append blank ولكنه يستخدم الامر replace للتعديل فى محتويات السجل الذى يقف عنده المؤشر

- Edit – program

```
set color to bg+/b,gr+/rb
clear
V_name = name
V_Phone = telephone
@ 1,1 say "Edit RECORD"
@4,2 say "Name : " GET v_name
@6,2 say "Phone :" GET v_phone
Read
@8,2 say "Saving Data ?"
@19,2 prompt "Yes"
@19,20 prompt "No"
menu to yn
if yn = 1
  replace name with v_name
  replace telephone with v_phone
endif
```

وايضا هذا البرنامج يفترض ان ملف البيانات مفتوح وان المؤشر يقف عند السجل الذى نريد تعديله



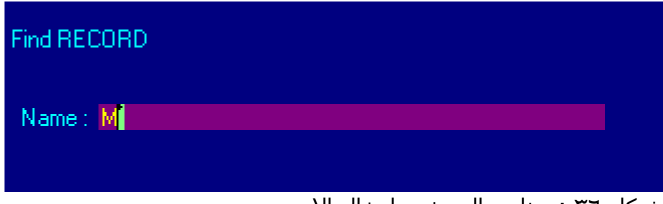
شكل ٣٥ : برنامج التعديل

برنامج البحث :

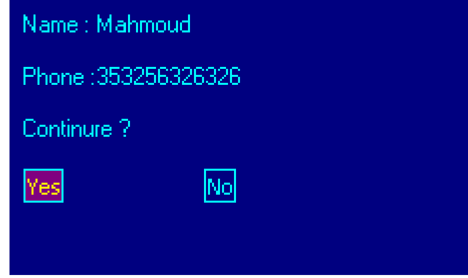
ان ابسط طريقة للبحث هى استخدام الامر locate والذى يبحث فى ملفات البيانات مباشرة (لا يبحث فى الملفات المفهرسة لذلك فهو ابطأ فى العمل) ويمكن استخدام الوظيفة ()found والتى تعطى القيمة .T. اذا تمت عملية البحث بنجاح وتمت الحصول على سجل يطابق شرط البحث ويمكن استخدام الامر Continue للاستمرار فى البحث والحصول على سجلات اخرى ينطبق عليها شرط البحث
يمكن استخدام الامر set exact off والذى يعنى عدم المطابقة وذلك فى حالة البحث الجزئى حيث نبحث مثلا فى عبارة حرفية مكونة من ٥٠ حرف من خلال عبارة اخرى عشرة حروف فقط فيطابق العشرة حروف الموجودة فى العبارة الكبيرة مع العبارة الصغيرة فقط .
ويمكن ان يحتوى شرط البحث على وظائف للتحكم فى العبارة مثل str و val و substr وغيرها والتى تساهم فى بناء التعبيرات الازمة للحصول على النتائج المطلوبة بسهولة

- Find – program

```
set exact off
set color to bg+/b,gr+/rb
clear
v_name = space(50)
@ 1,1 say "Find RECORD"
@4,2 say "Name : " GET v_name
read
locate for name = alltrim(v_name)
if found()
do while found()    && استمر طالما يوجد سجلات
clear
@4,2 say "Name : " + name
@6,2 say "Phone :" + telephone
@8,2 say "Continure ?"
@10,2 prompt "Yes"
@10,20 prompt "No"
menu to yn
if yn = 1
 continue && عملية بحث اخرى
else
  exit  && خروج من الحلقة
endif
enddo
else
@8,2 say "Record Not Found"
endif
```



شكل ٣٦ : برنامج البحث – ادخال الاسم



شكل ٣٧ : برنامج البحث – الاستمر فى البحث عن سجلات اخرى

برنامج الحذف :

ان ابسط طريقة للحذف هى استخدام الامر Delete يليه Pack (البرنامج لمستخدم واحد فى نفس الوقت) وهناك طريقة اخرى وهى استخدام delete فقط ووضع الامر set delete on فى بداية البرنامج (ووضع الامر pack ضمن خيارات البرنامج) وبذلك يمكن استرجاع السجلات المحذوفة فى اى وقت باستخدام الامر Recall او التخلص منها نهائيا باستخدام الامر pack

ملحوظة : الدلة deleted() تعطى القيمة .T. اذا كان السجل معلم لغرض الحذف وهذه الدالة هامة فى حالة التعامل مع السجلات المحذوفة وهى متواجدة مختلطة مع السجلات غير المحذوفة (اى فى حالة set delete off)

- Delete – program

```
Set delete on  && لا تتعامل مع السجلات المعلمة لغرض الحذف
set color to bg+/b,gr+/rb
clear
@4,2 say "Name : " + name
@6,2 say "Phone :" + telephone
@8,2 say "Delete Record?"
@10,2 prompt "Yes"
@10,20 prompt "No"
menu to yn
if yn = 1
 delete
endif
```

- Second Delete – program

```
set color to bg+/b,gr+/rb
clear
@4,2 say "Name : " + name
@6,2 say "Phone :" + telephone
@8,2 say "Delete Record?"
@10,2 prompt "Yes"
@10,20 prompt "No"
menu to yn
if yn = 1
 delete
 pack
endif
```

- Recalling Deleted Record – program

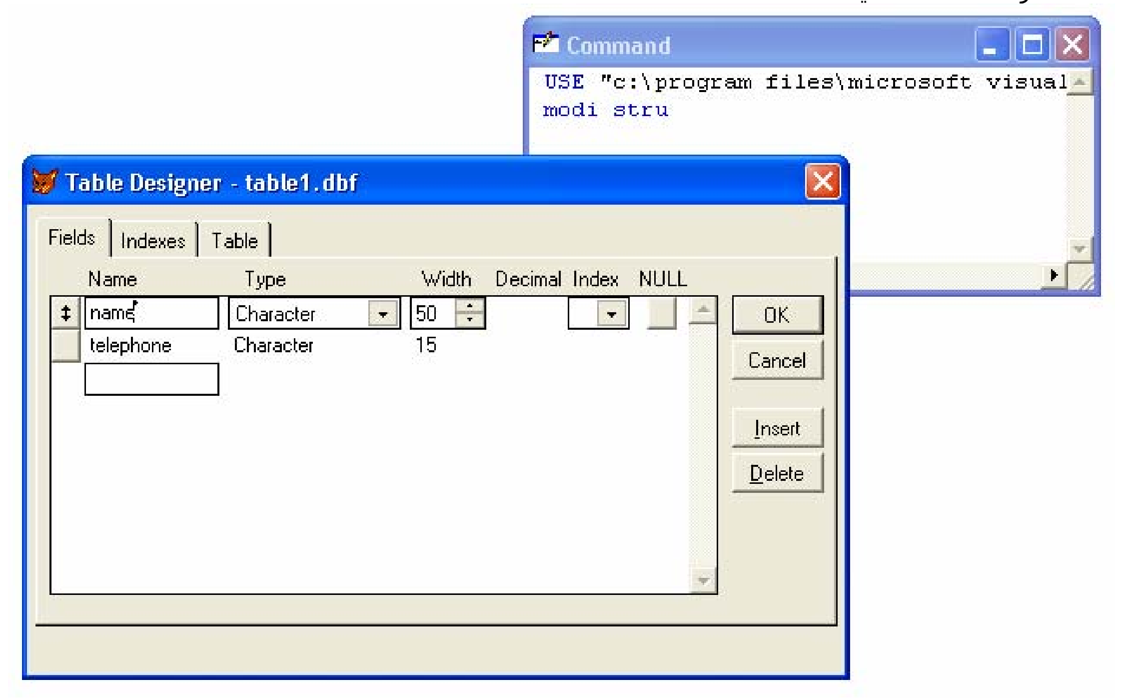
```
set color to bg+/b,gr+/rb
clear
If  deleted()
@4,2 say "Name : " + name
@6,2 say "Phone :" + telephone
@8,2 say "Resore Record?"
@10,2 prompt "Yes"
@10,20 prompt "No"
menu to yn
if yn = 1
 Recall   && استرجاع السجل المحذوف
endif
```

- Recalling All Deleted Records – program

```
set color to bg+/b,gr+/rb
clear
@8,2 say "Recalling all deleted  Records?"
@10,2 prompt "Yes"
@10,20 prompt "No"
menu to yn
if yn = 1
Recall All   && استرجاع كافة السجلات المحذوفة
Endif
```

تعديل مواصفات ملف البيانات :

ان هذه العملية يمكن ان تتم ببساطة شديدة
١- افتح ملف البيانات من خلال الامر use
٢- ادخل الامر modify structure او يمكن كتابته modi stru
٣- سوف تحصل على شاشة تعديل مواصفات الجدول (عدلها كما تشاء)
٤- اختار ok لحفظ التعديلات



شكل ٣٨ : تعديل مواصفات ملف البيانات (الجدول)

فتح اكثر من ملف بيانات :

ان فتح اكثر من جدول فى نفس الوقت امر بسيط من خلال فهم ما يسمى بمنطقة العمل work area حيث انك عندما تقوم بفتح جدول من خلال الامر use فانه يتم اسناد منطقة له ولتكن المنطقة 1 واذا قمت بفتح ملف اخر فانه يتم غلق الاول وفتح الاخر لانك تعمل فى نفس المنطقة
ولفتح اكثر من ملف فى نفس الوقت لابد من فتح كل ملف فى منطقة خاصة به
ويمكن اختيار منطقة العمل من خلال الامر select يليه رقم المنطقة
ويمكن تحديد المنطقة قبل الامر use وبالتالى نحدد اى منطقة بالتحديد يتم فتح الملف بها
او نستخدم الامر select بالصورة select 0 والتى تعنى الانتقال الى المنطقة التالية فمثلا لو كنت فى المنطقة ١ ينتقل الى المنطقة ٢ وهكذا
مثال : نفرض ان لدينا خمس جداول t1,t2,t3,t4,t5 (ملفات بيانات من t1.dbf,t2.dbf,..etc) ونريد فتحها معا

```
Use t1  && افتح الجدول فى المتطقة الاولى
Select 0 && انتقل الى المنطقة التالية وهى رقم ٢
Use t2
Select 0 && انتقل الى المنطقة التالية وهى رقم ٣
Use t3
Select 0
Use t4
Select 0
Use t5
```

ويمكن كتابة الاوامر على الصورة

```
Select 1
Use t1
Select 2
Use t2
Select 3
Use t3
```

```
Select 4
Use t4
Select 5
Use t5
```

وهكذا نستخدم امر select بالصورة التى تناسبنا لفتح الملفات
ولتحديد اى ملف نعمل معه نستخدم ايضا الامر select يليه رقم المنطقة المفتوح فيها الملف
مثال :

```
Select 3 && اختار المنطقة رقم ٣
Browse
```

وفى حالة عدم معرفة المنطقة يمكن تحديدها من خلال اسم الملف المفتوح بها كالتالى

```
Select t3
Browse
```

واثناء فتح الملف فى البداية بالامر use يمكن تحديد اسم رمزى للاشارة اليه

```
Use t1 alias cust
```

وهكذا يمكن تحديد t1 كالتالى

```
Select cust
```

وللتعامل مع بيانات الحقول من اكثر من منطقة يمكن استخدام السهم -> والذى ينطلق من اسم الجدول ويذهب الى اسم الحقل

مثال : نحن فى منطقة الجدول t2 ونشير لحق فى الجدول t3 اسمه name وحقل code فى الجدول t1 المسمى رمزيا cust

```
Select t2
? t3->name
? cust->code && او t1->code كما تريد
```

وهكذا يمكنك ببساطة التعامل مع اكثر من ملف فى نفس الوقت وتداول البيانات بينهما

البرمجة لاكثر من مستخدم :

عند كتابة البرامج لكى تعمل على الشبكة LAN (LOCAL AREA NETWORK) فانه تظهر الحاجة لاتاحة التعامل مع نفس البيانات من قبل اكثر من مستخدم وذلك يشترط وضع عدة اعتبارات فى الذهن اثناء البرمجة

١ – يجب فتح ملفات البيانات بحيث تكون قابلة للمشاركة SHARED

```
USE MYDATAFILE SHARED
```

٢ – يجب غلق ملف البيانات عند اضافة سجل جديد

```
IF FLOCK()
   APPEND BLANK
  ....CODE
ENDIF
```

٣ – يجب غلق السجل عند تعديل بياناته

```
IF RLOCK()
  ....CODE
ENDIF
```

٤ – عند الانتهاء من العملية التى تستلزم غلق السجل او الملف ينبغى فتح الملف او السجل وفى نفس الوقت نستدعى الوظيفة TABLEUPDATE(.T.) كى يظهر اثر التعديلات التى قمنا بها على ملفات البيانات

```
UNLOCK
TABLEUPDATE(.T.)
```

٥ – عند تعديل اكثر من سجل معا ينبغى غلقهما معا.

- ولفتح الملف بحيث تكون هناك امكانية المشاركة فى البيانات نضيف الكلمة SHARED فى امر USE .
- ولاغلاف الملف نستخدم الوظيفة FLOCK() والتى تعطى القيمة TRUE اذا تم اغلاق الملف بنجاح .
- والدالة RLOCK() مثل FLOCK() الا انها تقوم باغلاق سجل واحد فقط والذى يقف عنده الموشر.
- الامر UNLOCK يعيد فتح الملف او السجل المغلق

ملاحظة : هناك اوامر تشترط فتح الملف بصفة خاصة EXCLUSIVE مثل الامر ZAP و PACK

```
USE MYDATAFILE EXCL
ZAP
```

- الامر SET EXCLUSIVE OFF يجعل كل الملفات التى يتم فتحها بالامر USE تكون SHARED فى حين SET EXCLUSIVE ON يجعل كل الملفات التى يتم فتحها باستخدام الامر USE تكون EXCLUSIVE

ملحوظة : عند فتح ملف البيانات على الشبكة ينبغى ان يكون الدليل FOLDER الذى يحتوى على الملف SHARED وينبغى ادخال المسار بدقة مثل

```
SET EXCLUSIVE OFF
USE \\SERVER\MYDATA\TABEL1.DBF
```

نسخ البيانات من ملف لاخر :

عند كتابة البرامج التى تعتمد على نظام للملفات (مثل برنامج Microsoft word الذى يحتوى على نظام خاص للملفات من نوع DOC) داخل فيجوال فوكس برو فان افضل طريقة ان يكون هذا الملف من النوع DBF. وتقوم بتغير الامتداد الخاص به من DBF. الى الامتداد الذى يناسبك وليكن مثلا MYE.
وعندها تحتفظ بصورة من الملف فارغا(فى صورته الاولى) (ملف DBF. وقد يحتوى على عدد من السجلات الازمة لتحديد مواصفات الملف الجديد) وعند الحاجة لانشاء ملف جديد فانك تنسخ بيانات الملف (DBF.) الى الملف (MYE.) الجديد

ولنسخ البيانات من ملف لاخر نستخدم الامر COPY TO

والمثال التالى هو ابسط صورة لاستخدام هذا الامر

```
Mynewfile = "myfile.MYE"
USE MYDBFFILE
COPY TO (Mynewfile)              && can be use like : copy to myfile.mye
```

وفى هذا المثال تم اعلان متغير حرفى اسمه mynewfile يحتوى على اسم الملف الجديد المطلوب انشائه وهو myfile.mye وهذا الملف سوف ياخد مواصفاته من ملف البيانات mydbffile.dbf

العلاقات بين الجداول :

هى عملية بسيطة اذا تم فهمها بامثلة جيدة من الواقع – تخيل انك تقوم بعمل برنامج لمتابعة حسابات Super Market وان برنامجك مسئول عن اصدار فواتير للزبائن بحيث تحتوى هذه الفاتورة على رقم يميزها بالاضافة الى ما قام الزبون بشرائه من الاصناف المتاحة للبيع – عملية البرمجة هنا مؤكد انها بحاجة لملف بيانات لتخزين بيانات الفاتورة – ولكن المشكلة فى بيانات الاصناف – حيث انك لو وضعت فى الفاتورة عشرة حقول مثلا لبيانات الاصناف (صنف١ – سعر١ – خصم١ – صنف٢ – سعر٢ – خصم٢ وهكذا ....صنف١٠ – سعر ١٠ – خصم ١٠)
فسوف تجد انك قمت بعمل ٣٠ حقل فى ملف البيانات من اجل تخزين بيانات عشرة اصناف.
واذا اشترى الزبون ٥ اصناف تكون هناك العديد من الحقول الفارغة بدون داعى – واذا اشترى ١٠ اصناف فهى حالة مثالية نادرة الحدوث

ولكن ماذا سوف تفعل لو اشترى الزبون ١١ صنف ؟

اذا اشترى الزبون ١١ صنف سوف تكون هناك مشكلة حيث ان البائع سوف يقوم بعمل فاتورتين من اجل حل المشكلة مع الزبون ولكنه سرعان ما سوف يتصل بك لايجاد حل
سوف تبتسم بالطبع وتقول حصل خير وتقوم بزيادة عدد الحقول لتستوعب اصناف اكثر وليكن **٢٠ ومن هنا تبد لعبة القط والفار اذا جاء زبون يشترى اكثر من ٢٠ صنف !**

ان الحل لتلك المشكلة هو ان تقوم بانشاء ملف مستقل للاصناف ويحتوى هذا الملف على الحقول (رقم الفاتورة – رقم الصنف – السعر – الخصم)
وتقوم بانشاء ملف اخر للفواتير يحتوى على (رقم الفاتورة – تاريخ الفاتورة) وملف اخر لاسماء الاصناف (رقم الصنف – اسم الصنف)

وتبدا بعمل علاقات بين تلك الجداول حيث تربط بين ملف الفاتورة وملف بيانات الاصناف عن طريق رقم الفاتورة (لاحظ انه يتم الربط من خلال حقل بنفس الاسم والنوع) وتربط بين ملف الاصناف وبيانات الاصناف من خلال (رقم الصنف)

ولعمل علاقة بين جدولين نستخدم الامر SET RELATION TO

مثال

```
SELECT Invoice
SET RELATION TO Inv_ID INTO InvItems
```

فى هذا المثال تم عمل علاقة بين ملف الفاتورة Invoice وملف بيانات الاصناف InvItems عن طريق رقم الفاتورة Inv_ID .

ومعنى هذه العلاقة انه عند انتقال الموشر من سجل لاخر داخل ملف الفاتورة Invoice فانه يتم تصفية ملف الاصناف InvItems بحيث يظهر كانه يحتوى فقط على السجلات التى بها يكون رقم الفاتورة InvItems هو نفسه رقم الفاتورة التى يقف عندها الموشر فى ملف الفاتورة

ملحوظة : يشترط لعمل علاقة بين جدولين عن طريق حقل بيانات – ان يكون ملف البيانات مفهرس تبعا لبيانات هذا الحقل ويمكن تحديد الحقول التى يتم فهرسة البيانات تبعا لها اثناء انشا ملف البيانات او من خلال الاوامر كالتالى :

```
USE Invoice
Index ON Inv_ID TO MyIndex
```

قاعدة البيانات Database والاستعلام وجمل SQL :

فى الواقع انا من الذين يحبون الاشادة بالاعمال الجيدة للاخرين – لهذا فان كتاب قواعد فيجوال فوكس برو – تاليف السيد محمد الهدهد قد قام بتغطية هذه الجزئية بصورة جيدة وهذا الكتاب مجانى ويوجد فى موقع www.cb4a.com ولهذا يجب قراءته قبل استكمال التعامل مع البيانات البعيدة Remote Data ويجب ايضا ان تتعلم من الكتاب كيفية انشاء التقارير وملفات المشاريع وكيفية عمل برنامج setup لبرامجك لان هذا الكتيب لن يشمل تلك المواضيع لمراعاة عدم التكرار.

التعامل مع البيانات البعيدة Remote Data:

من اهم مفاهيم برمجة قواعد البيانات الان برمجة الزبون-خادم (Client-Server) وخاصة مع قاعدة البيانات SQL Server والتى تتسم بالانتشار الكبير والقوة الهائلة والسرعة العالية.
وتتيح لغة برمجة فيجوال فوكس برو ان تقوم بعمل Interface واجهة للبرنامج من خلال اللغة والتى تقوم بتخزين البيانات من خلال التعامل مع SQL Server
وبالطبع للعمل فى البرمجة بتلك الطريقة ينبغة الالمام الجيد بطريقة استخدام SQL Server وكذلك فهم لغة الاستعلام SQL والمثال التالى هو مثال بسيط يوضح كيفية الاتصال بقاعدة البيانات SQL Server من خلال اوامر فيجوال فوكس برو باستخدام تقنية ODBC مع SQL path through :

```
open database mydata
CREATE CONNECTION mahcon DATASOURCE "MAHSQL" USERID "<mah>" PASSWORD "<msfclipper>"
myh = SQLCONNECT("mahcon","MAH","msfclipper")
SQLEXEC(myh,"select * from t1","myt1")
browse
sqldisconnect(myh)
DELETE CONNECTION mahcon
```

وفى هذا المثال يتم اولا فتح قاعدة البيانات Mydata والتى تحتوى على العديد من الجدوال ثم بعد ذلك يتم انشاء connection اى اتصال وهو الازم للاتصال بقاعدة البيانات SQL Server وقد سميناه mahcon ويلى ذلك بيانات الاتصال من حيث مصدر البيانات DATASOURCE ويتم تعريفه من خلال لوحة التحكم (فى WINDOWS) ويتم اعطاء اسم المستخدم وكلمة السر
بعد ذلك ستم استخدام جمل SQL Path through وهى عبارة عن مجموعة من الدوال تبدا ب SQL لسهولة التعامل مع SQL Server
الدالة ()SQLCONNECT تبدا عملية الاتصال من خلال تحديد اسم الـ Connection واسم المستخدم وكلمة السر وهذه الدالة ترجع قيمة handle يستخدم بعد ذلك لارسال تعليمات من خلاله.
الدالة ()SQLEXEC تقوم بتنفيذ جمل الاستعلام SQL وترجع النتائج فى Cursor تخيلى (ملف بيانات تخيلى) وهنا فى هذا المثال myt1
الدالة ()SQLDISCONNECT تقوم بالغاء الاتصال.
الامر Delete connection لحذف الاتصال الذى قمنا بانشاءه داخل قاعدة البيانات database واذا لم نقم بذلك واستخدمنا نفس التعليمات مرة اخرى والتى تنشى الاتصال من جديد سوف تحصل على رسالة بان الاتصال موجود من قبل.

طريقة اخرى باستخدام ADO (ACTIVEX DATA OBJECT)

```
LOCAL myobj,myobj2,myname
myobj = CREATEOBJECT("ADODB.CONNECTION")
MYOBJ.connectionstring = "Provider=SQLOLEDB.1;data source=(local);Persist Security Info=False;User ID=MAH;Initial Catalog=MAHTEL;PASSWORD=msfclipper"
myobj.open
myobj2 = CREATEOBJECT("ADODB.RECORDSET")
myobj2.open("SELECT * from t1",myobj)
myname = myobj2.fields.item(0).value
```

وفى هذا المثال تم عمل اتصال connection ثم recordset ثم الحصول على قيمة الحقل الاول فى الجدول t1 والذى يسمى NAME ووضعها فى متغير MYNAME
ولفهم هذا المثال انت بحاجة لدراسة ADO بالاضافة الى معرفة مفهوم الكائنات OBJECTS.



الباب الثالث

برمجة الكائنات

مقدمة هامة :

فى الواقع ان برمجة الكائنات هى متعة البرمجة اذا فهمت جيدا ماهى – واغرب مفاهيمها اذا لم تكن تدرك مدى فائدتها واهميتها

س : انا مبرمج محترف واقوم بعمل برمجيات متكاملة للسوق ولا اعرف ما هى برمجة الكائنات ولا احتاجها فى عمل برامجى – فهل هى مهمة ؟؟

جـ : للاسف هذا السوال المتناقض منتشر بين عدد كبير من المبرمجين الذين بالفعل يجيدون عمل برامج تجارية يمكن بيعها فى الاسواق.
ان سبب التناقض فى السوال ان السائل لا يعرف ما هى برمجة الكائنات ويقول انه لا يحتاجها فى عمل برامجه – والرد على ذلك التناقض – **كيف تنفى حاجتك لشىء لا تعرف ما هو !**
فمثلا اذا كان لديك سيارة حديثة وبحالة جيدة ويمكن ان تلف بها العالم كله فان هذا لا يكون مبررا لان تنفى حاجتك لان يكون لديك طائرة اذا كنت لا تعرف ما هى الطائرة !

س : هل حقا برمجة الكائنات ينطبق عليها مثال السيارة والطائرة ؟

جـ : فى الواقع لا لاننى اذا قلت نعم فانى اظلم برمجة الكائنات والتى يمكن القول بان الفرق بينها وبين البرمجة التقليدية كالفرق بين السلاح النووى والمبيد الحشرى !

س : هل يعنى ذلك ان استخدام نمط برمجة الكائنات فى برامجى امر ضرورى ؟

جـ : بصراحة لو فهمت المثال جيدا تعلم انها ليست قاعدة عامة بمعنى انه من غير التعقل ان تقوم بقتل ذبابة باستخدام سلاح نووى – ولكنه يلزمك فى المعارك الضارية

س : انا مبرمج جيد وكلما قرات كتاب عن برمجة الكائنات احس بانى لم استفد شيئا فى الناحية العملية فهل من طريق مناسب لحل تلك المشكلة ؟

جـ : ابتسم ياعزيزى لانك من خلال هذا الكتيب سترى اسلوبا مختلفا لفهم برمجة الكائنات – فانت معى لن تتعلمها بصورة مباشرة كما فى جميع الكتب – **ولكننا سنقوم بجولة مثيرة سوف تجعلك تخترع معى نمط برمجة الكائنات بنفسك ولكن تذكر انه موجود فهو ليس اختراع وانما اكتشاف.**

ملحوظة : ان فيجوال فوكس برو من اللغات النادرة التى تدعم برمجة الكائنات بنسبة مئة بالمئة وفى نفس الوقت توفر انماط اخرى مثل البرمجة الهيكلية و ادارة الاحداث والجداول العلاقية الموجهة والقالب.

العالم قبل برمجة الكائنات :

لقد كان سعيدا بالبرمجة الهيكلية لدرجة كبيرة لانها فى حد ذاتها كانت قفزة نوعية ومع انتشار الحاسبات الشخصية وتعدد البرمجيات وتزايد الطلب عليها ظهرة مشكلة كبيرة !

هل يمكن لابنى الصغير الذى يبلغ من العمر ٦ سنوات ان يعمل على الحاسب ؟

بالفعل كانت صدمة لعلماء الحاسب – ماذا قدمنا نحن للاطفال ؟ - اليس من حقهم التعامل مع تلك الالة الذكية بلى من حقهم .
ولكن كيف يتعامل الطفل الصغير الذى لا يعرف سوى القليل من الكلمات (قراءة وكتابة) مع الحاسب.

انك اذا اردت ان تعرف طفلك على حيوانات الغابة فمن السهل ان تصطحبه الى حديقة الحيوانات ليعرف اشكال الحيوانات وطبيعتها واصواتها – ومن الحلول الاسرع ان تحضر كتيبا يشمل العديد من الصور الفوتغرافية لسهولة التعرف على الحيوانات او لروية اشياء لا يمكن الوصول اليها مثل الاسماك المتوحشة كالحوت فى عالم البحار

ومن هنا كانت فكرة عمل برامج رسومية على الحاسب تستخدم فى تعليم الاطفال .

وفى المراحل الاولى من عمل بيئة رسومية كواجهة للمستخدم الذى كان الطفل الصغير لم يكن الامر منتشرا على مستوى الحاسب بصفة عامة – فقد كانت الشاشات النصية هى الملاذ الوحيد كواجهة للبرامج التجارية وانظمة المعلومات.

والاسباب التى تاتى وراء عدم انتشار البيئة الرسومية :-
١ – صعوبة شديدة فى برمجة الجرافك – فما بالك ببرمجة بيئة رسومية متكاملة
٢ – التكلفة العالية والتى بحاجة للعديد من المبرمجين المحترفين
٣ – الروية الضيقة لبعض الذين ظنوا انها ضرورية للاطفال فقط

اريدك وانت تقرا هذا التاريخ ان تنظر نظرة بسيطة على الجرافك والبيئة الرسومية التى تنعم بهم الان وتفكر قليلا ولاول مرة تسال نفسك **(هل هذا صعب – ياترى كم مبرمج وراء ذلك البناء)**
فى الواقع انه صعب ويقف وراء نظام النوافذ وندوز كاملا حوالى ٣٠٠٠ مبرمج (محترف جدا جدا .....الى ماشاء الله)

بدا فى تلك الفترة الماضية من الزمن العديد من العلماء فى التفكير لماذا برمجة البيئة الرسومية صعبة ؟ وكان رد العلماء على انفسهم ان الاسلوب المستخدم فى البرمجة (البرمجة الهيكلية) غير مريح ويسبب العديد من الارباك

س : كيف يكون اسلوب البرمجة الهيكلية غير مريح – اريد مثالا على ذلك ؟

جـ : حسنا – اظن انك الان تقرا الكتاب على الحاسب اذا لم تكن قد قمت بطباعته – هذا يعنى انك تعمل على برنامج اكروبات ريدر – ببساطة قم بعمل تصغير لنافذة البرنامج (ليس الان – اكمل قراءة السطرين التالين اولا) وبقية البرامج التى تعمل جميعا قم بتصغيرها حتى تصل لسطح المكتب الذى غالبا ما يكون مزدحم بالايقونات وانظر الى كل ايقونة وتخيل لو انك قمت ببرمجة تلك العملية فقط – كيف ستبرمجها من الصفر ؟؟؟؟؟

انت تريد تنفيذ اكثر من عملية فى وقت واحد وهى تصغير النافذة – ثم عرض ايقونات سطح المكتب – تخيل كم مصفوفة انت بحاجة اليها لتخزين بيانات مثل هذا النظام المرن – تخيل كيف سوف تتعامل مع تلك المصفوفات ذات العدد الهائل والتى قد تصل الى اربعة ابعاد – تخيل كيف سوف يعمل عدد كبير من المبرمجين معا فى مشروع بهذا التعقيد.

هذا ما فكر فيه علماء البرمجة ووصلوا الى ان فكرة كون البرنامج مقسم بصورة هيكل (البرمجة الهيكلية) صورة غير كافية لتلائم التعقيد الموجود فى برنامج ادارة البيئة الرسومية

لما فكر العلماء فى المشكلة بحجمها الكامل لم يصلم لحل – ولهذا قررم التفكير فى المشكلة بعد تجزئها الى اجزاء فرعية وكانت تلك الاجزاء هى

١ – البيانات المشتركة التى تتعامل معها اجزاء النظام المختلفة وصلت الى كم هائل يصعب تنظيمه وادارته بين المبرمجين – ويصعب على المبرمج الواحد التعامل مع هذا الكم الهائل من البيانات
٢ – تكرار الحاجة لنفس العمليات بصور مختلفة اكثر من مرة وفى نفس الوقت تتعامل مع البيانات الهائلة (مثلا كل العناصر المكونة للبيئة الرسومية تتعامل مع مدخلات الفارة ولوحة المفاتيح ولكن كل يدير تلك المدخلات باسلوب مختلف) وفى البرمجة الهيكلية يصعب تنظيم ذلك بدون تداخل المسميات ونسخ الاكواد وغيرها من العمليات التى تربك النظام وتزيده تعقيدا
٣ – فهم النظام من قبل المبرمجين يحتاج وقت طويل جدا لا يمكن تحمله وصيانة النظام اصبحت امر صعب مما قد يدفع لاعادة كتابة اجزاء النظام من البداية بدلا من صيانتها لشدة التعقيد واختلاف المبرمجين

الاسباب الثلاثة هى عناصر المشكلة – والحل الذى نود ابتكاره هو بديل للبرمجة الهيكلية ولكن ما هذا البديل – المنطق يقول بان البديل هو المركب الذى يمثل مجموعة حلول للمشاكل المتوفرة .

اولا : حل مشكلة البيانات :

ان مشكلة البيانات فى البيئة الرسومية تنقسم الى مشكلتين
١ – مشكلة البيانات المتشابهة
٢ – مشكلة البيانات المشتركة
ان البرمجة الهيكلية تقدم حلا لهاتين المشكلتين – ولكنه حل قديم لا يتناسب مع المشكلة الكبيرة التى يواجهها العلماء فى تلك الفترة

١ – مشكلة البيانات المتشابهة :-

مثال لذلك ايقونات سطح المكتب فى وندوز والتى جميعا تحمل نفس السمات ولكن بخصائص مختلف السمات هى وجودها على سطح المكتب والذى يشترط احداثيات المكان وكذلك الصورة التى تمثل الايقونة وغيرها
والخصائص هى قيمة تلك السمات مثل قيمة الاحداثيات ومحتوى الصورة (سواء كانت مصفوفة او ملف صورة).

س : ما هو حل البرمجة الهيكلية لمشكلة البيانات المتشابهة ؟

جـ : الحل القديم ببساطة مصوفة متعددة الابعاد (كل عنصر ذو بعد خاص) تكفى لكى تحمل بيانات الايقونات

مشكلة الحل : التعقيد الناتج عن التعامل مع المصفوفات ذات الابعاد المختلفة.

س : ما هو حل برمجة الكائنات لمشكلة البيانات المتشابهة ؟
جـ : نشاء مفهوم جديد الفصيلة والتى سوف نسميها مثلا فصيلة الايقونات وهذه الفصيلة سوف نحدد لها سمات وهو الالزمة للايقونة مثل الاحداثيات وملف الصور وغيرها
وكلما نحتاج ايقونة جديدة ننشى مايسمى بالكائن والذى يحتوى على سمات الفصيلة ولكن بخصائص مختلفة

اى ان كل ايقونة على سطح المكتب هى كائن وجميع هذه الكائنات نشات من فصيلة واحدة هى فصيلة الايقونات
اى ان كل كائن لابد له من فصيلة والفصيلة هى مصدر الكائنات.

س : ما فائدة هذا الحل ؟
جـ : سهولة التعامل مع البيانات المتشابهة حيث انك تتعامل مع كل كائن من خلال اسمه وقد تتعامل معه من خلال الفصيلة التى تحتوى على موشر على الكائنات التى اصدرتها.
وفى نفس الوقت سهولة تعديل السمات التى تكون الفصيلة.

١ – مشكلة البيانات المشتركة :-
مثال على ذلك مقاس العرض الخاص بالشاشة ابعاد الشاشة فى النمط الرسومى – الشائع ٨٠٠* ٦٠٠
هذه مثلا معلومة عامة تحتاجها فى اى وقت عند كتابة اجزاء نظام البيئة الرسومية.

س : ما هو حل البرمجة الهيكلية لمشكلة البيانات المشتركة ؟
جـ : انها تقدم حلا بسيطا يكمن فى التحكم فى مدى انتشار المتغيرات على مستوى اجزاء النظام ولكن ذلك الحل لا يتناسب مع الالاف من المتغيرات العامة التى نحتاجها للسيطرة على النظام

س : ما هو حل برمجة الكائنات لمشكلة البيانات المتشابهة ؟
جـ : الحل هو ربط البيانات مع الاكواد التى تعمل عليها وفى حالة طلب بيانات خارجية يتم تقديم مفهوم التراسل بين الكائنات بمعنى كائن يطلب من كائن معلومة او وظيفة معينة فيقوم بتاديتها له.

س : ما فائدة هذا الحل ؟
جـ : تنظيم هائل منقطع النظير للبيانات حيث دائما تجد البيانات التى تعمل عليها الاكواد معا فى فصيلة واحدة وفى حين طلب بيانات خارجية فان فصيلة اخرى تتعامل مع ذلك الامر
وتجدر الاشارة بامكانية السمات على الفصيلة (سمات الفصيلة للفصيلة وليس للكائنات) وعندها فان هذه السمات تكون منها نسخة واحدة للفصيلة كلها ولا يتم عمل نسخ منها عند انشاء كائنات جديدة

ثانيا: حل مشكلة التعليمات المتشابهة :
مثلا فى البيئة الرسومية يكون هناك العديد من النوافذ التى تشمل العديد من العناصر المختلفة (زر امر – مربع نص – قائمة خيارات وغيرها الكثير) وجميع هذه العناصر تمتلك اكواد متشابهة عند برمجتها للتعامل مع احداث الفارة والماوس مع وجود اختلافات بينها فى كيفية تداول تلك المعلومات القادمة من وحدات الادخال

فمثلا لو ضغط بزر الفارة الايسر على مربع نص فان الموشر يستعد لاستقبال بيانات من لوحة المفاتيح ولكن اذا ضغط على زر امر فان يقوم بتنفيذ التعليمات المخزنة له بخصوص هذا الحدث .

الحدث مشترك بينها لكن الفعل مختلف .

س : ما هو حل البرمجة الهيكلية لمشكلة التعليمات المتشابهة ؟
جـ : كتابة وظائف او دوال مختلفة لتنفيذ المهة وهذا حل جيد غير مرن عندما نجد اننا نحتاج كتابة الالاف من الدوال التى تتداخل فى المسميات وتختلف اختلاف بسيط فى المهام ويزداد التعقيد اذا كانت تلك الدوال تنادى بعضها بكثرة فنكون اشبه بالغارق داخل ماكرونة اسباكتى.

س : ما هو حل برمجة الكائنات لمشكلة التعليمات المتشابهة ؟
جـ : الحل تقديم مفهوم جديد وهو الوراثة بمعنى عمل فصيلة تحتوى على العمليات المشتركة بالاضافة الى السمات التى تحتاجها هذه العمليات ثم عندما نحتاج حالة اخرى مثلا فصيلة زر امر ننشئها من هذه الفصيلة الشائعة وعندها تحتوى فصيلة زر الامر على كل خصائص الفصيلة الشائعة ثم نضيف بعد ذلك السمات المختلفة

س : ما فائدة هذا الحل ؟
جـ : توفير اعادة كتابة التعليمات والحفاظ على التنظيم العالى وسهولة التعامل بنفس المسميات مثل حدث Click يظل بنفس الاسم فى مختلف الفصائل بدون اى مشاكل

برمجةالكائنات فى فيجوال فوكس برو :-

فى هذا الجزء سوف نتعلم :-

١ – انشاء الفصائل
٢ – انشاء كائنات من الفصيلة
٣ – كتابة دوال داخل الفصيلة
٤ – التعامل مع سمات الفصيلة من خلال الدوال
٥ – الوراثة
٦ – التشابه
٧ – الفصائل الاساسية داخل اللغة
٨- برمجة واجهة البرنامج من خلال الفصائل الاساسية
٩- طريقة اعداد التطبيقات التجارية باسلوب الصفوف الثلاثة

اولا : امثلة توضح الاساسيات :-

انظر الى المثال التالى والذى ينشى فصيلة جديد باسم الهاتف وتحتوى على ثلاث متغيرات هما الاسم والعنوان والتليفون

```
* program1.prg
DEFINE CLASS mytelephone as Custom
  myname = SPACE(50)
  myaddress = SPACE(50)
  mytelephone = SPACE(15)
  PROCEDURE showdata()
  SET COLOR TO w/b
  CLEAR
  @2,2 say this.myName
  @4,2 say this.myaddress
  @6,2 say this.mytelephone
  RETURN
ENDDEFINE
```

لا داعى للارتباك فان الامر فى غاية البساطة – بداية قم بانشاء اجراء جديد (ملف برنامج program1.prg) بحيث يحتوى على البرنامج السابق

فى هذا البرنامج تم تعريف فصيلة جديدة من خلال الامر define class يلى ذلك الامر اسم الفصيلة الجديدة ثم كلمة as يليها اسم الفصيلة الام والتى تم اختيارها لتكون custom

نلاحظ ان محتويات الفصيلة يتم تحديدها بعد جملة DEFINE وتنتهى قبل جملة ENDDEFINE

تم تحديد سمات الفصيلة بحيث تحتوى على ثلاث متغيرات حرفية بالاضافة الى اجراء واحد هو SHOWDATA() وينبغى معرفة ان هذا الاجراء اصبح له اسم اخر عندما يكون داخل الفصيلة وهذا الاسم هو Method

يحتوى هذا الاجراء على مفهوم جديد وهو استخدام الكلمة this للوصول الى السمات داخل الفصيلة

وعند تنفيذ ملف البرنامج program1.prg فانك لم تر نتيجة لتنفيذه

والان انظر الى المثال التالى :

```
* program2.prg
SET PROCEDURE TO program1
myobj = CREATEOBJECT("mytelephone")
myobj.myname = "mahmoud samir fayed"
myobj.myaddress = "jeddah"
myobj.mytelephone = "6295881"
myobj.showdata()
```

هذا المثال سوف يعطى نتيجة منظورة – حيث انه فى البداية يشير الى ملف اخر هو program1.prg وذلك يسمح للبرنامج program2.prg باستخدام الفصيلة المعرفة سابقا فى الملف program1.prg

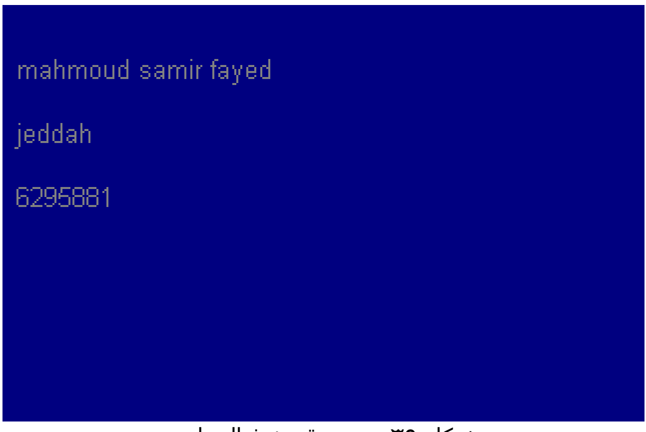
ثم بعد ذلك ينشى البرنامج كائن جديد باسم Myobj من الفصيلة mytelephone باستخدام الدالة createobject() والتى تستخدم لهذا الغرض (انشاء كائنات جديدة من الفصائل المعرفة سابقا)

وبعد ذلك يصبح لدينا كائن جديد يحمل الاسم Myobj وله سمات الفصيلة وهى الثلاثة متغيرات بالاضافة الى الاجراء showdata()

يحدد البرنامج الان خصائص الكائن من خلال العلامة (.) dot والتى تستخدم لهذا الغرض (التعامل مع سمات الكائن التى اخذها من الفصيلة سواء كانت متغيرات او دوال)

وبعد اسناد القيم ينادى البرنامج الدالة showdata()

ونتيجة تنفيذ البرنامج كالتالى :



شكل ٣٩ : نتيجة تنفيذ البرنامج

والان يتضح لنا انه يمكن الاستفادة من الفصيلة من خلال توليد كائنات منها – وانه يمكن انشاء العديد من الكائنات من فصيلة واحدة ومع ذلك تظل سهولة التعامل والفهم متوفرة فى تعليمات البرنامج .

والمثال التالى يوضح ذلك

```
* program3.prg
SET PROCEDURE TO program1
myobj  = CREATEOBJECT("mytelephone")
myobj.myname = "mahmoud samir fayed"
myobj.myaddress = "jeddah"
myobj.mytelephone = "6295881"
myobj2 = CREATEOBJECT("mytelephone")
myobj2.myname = "ahmed samir fayed"
myobj2.myaddress = "Egypt"
myobj2.mytelephone = "3350641"
myobj.showdata()
wait
myobj2.showdata()
```

حيث قمنا بانشاء كائنين myobj & myobj2 من فصيلة واحدة هى Mytelephone
ونتيجة تنفيذ هذا المثال هو عرض بيانات myobj ثم myobj2 .

ثانيا : مفهوم الوراثة :-

هو اهم مفاهيم برمجة الكائنات وينبغى معرفة ان الانظمةاو البرامج المبنية على برمجة الكائنات ولا تحتوى فى داخلها على الوراثة لا يقال عنها انها (نظام برمجة كائنات) – وانما يقال(نظام بمساعدة برمجة الكائنات)

الوراثة ببساطة هى انشاء فصيلة جديدة من خلال سمات فصيلة سابقة موجودة من قبل بحيث تحمل الفصيلة الجديدة كل سمات الفصيلة الموجودة سابقا مع اجراء بعض التعديلات او الاضافات على سمات الفصيلة سواء كانت متغيرات او دوال.

والمثال التالى يوضح ذلك حيث نقوم بانشاء فصيلة جديدة تسمى person وهذه الفصيلة تحمل نفس سمات الفصيلة mytelephone ولكن يضاف اليها متغيرات جديدة كما يتم تعديل الاجراء showdata() بداخلها بحيث يعرض السمات الجديدة كاملة

```
* program4.prg
SET PROCEDURE TO program1
DEFINE CLASS myperson as mytelephone
  mygender = SPACE(50)
  mydateofbirth = SPACE(10)
  PROCEDURE showdata()
  mytelephone::showdata()
  @8,2 say this.mygender
  @10,2 say this.mydateofbirth
  RETURN
ENDDEFINE
```

فى هذا المثال نلاحظ كيف نعرف فصيلة جديدة من فصيلة سابقة وذلك بكتابة اسم الفصيلة السابقة (الفصيلة الام او الفصيلة الاساسية) بعد كلمة as

ثم نعرف المتغيرات الجديدة – ونلاحظ اعادة تعريف الاجراء showdata() مرة اخرى

ونلاحظ ان الاجراء showdata() ينادى الاجراء showdata() الموجود فى الفصيلة الام وذلك بكتابة اسم الفصيلة الام ثم (::) يليها اسم الاحراء فى الفصيلة الام

وبالتاكيد اذا قمت بتنفيذ هذا البرنامج فانك لن ترى نتيجة مباشرة

المثال التالى يعطى نتائج

```
* program5.prg
SET PROCEDURE TO program4
myvar = CREATEOBJECT("myperson")
myvar.myname = "mahmoud samir fayed"
myvar.myaddress = "jeddah"
myvar.mytelephone = "6295881"
myvar.mygender = "male"
myvar.mydateofbirth = "29/12/1986"
myvar.showdata()
```

ونتيجة تنفيذ هذا المثال هى :



شكل ٤٠ : نتيجة تنفيذ البرنامج

ونلاحظ بعد هذه الامثلة ما تضيفه الوراثة من حل لمشاكل كثيرة فهى توفر عليك تكرار الاكواد الصعبة وتتيح فرصة اكبر لكى يكون البرنامج منظما.

ملحوظة : يمكن عمل وراثة اكثر من مرة بمعنى ان الفصيلة myperson والتى نشات من الفصيلة mytelephone يمكن ان ننشى منها فصيلة اخرى ولتكن Myemployee وعندها تصبح الفصيلة myperson هى الفصيلة الام وذلك فى علاقتها مع الفصيلة Myemployee فى حين انها الفصيلة الابن وذلك فى علاقتها مع الفصيلة Mytelephone

ثالثا : مفهوم التشابه -:

ببساطة ان الفصائل المختلفة يمكن ان تحتوى على سمات (متغيرات او دوال) بنفس الاسم فمثلا كل من الفصيلة mytelephone والفصيلة Myperson يشملان بداخلهما اجراء بنفس الاسم هو showdata() وهذا يسهل التعامل مع الكائنات المتولدة من فصائل مختلفة.

رابعا : الفصائل الاساسية داخل اللغة -:

من اروع ما تمتلكه لغة فيجوال فوكس برو مجموعة هائلة من الفصائل الاساسية المعرفة مسبقا من قبل مطورى اللغة بحيث يمكن استخدام هذه الفصائل او اعادة صياغتها من خلال انشاء فصائل جديدة
وهذه الفصائل الاساسية تخدم الاغراض المتعددة مثل واجهة البرنامج وادارة البيانات وغيرها الكثير .

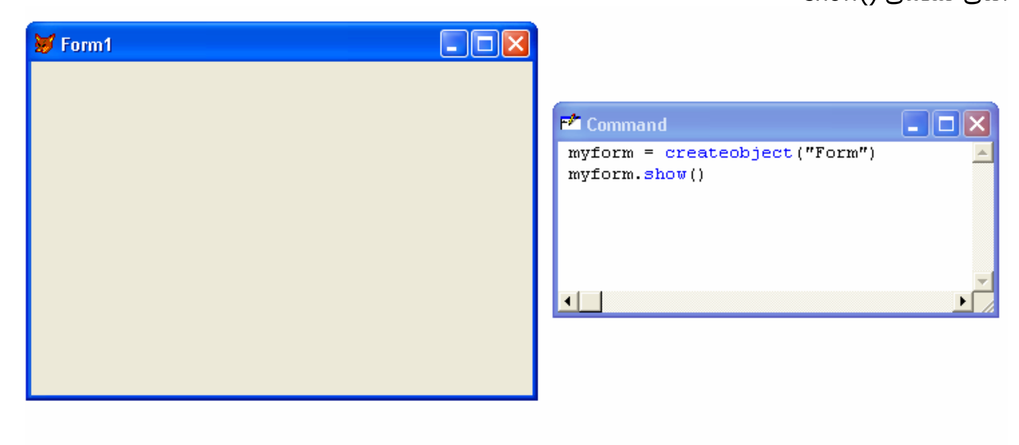
خامسا : برمجة واجهة البرنامج من خلال الفصائل الاساسية -:

اعلم ان ٩٠% من التطبيقات التى تحتوى على واجهة رسومية لا يتم برمجتها من خلال الاكواد او التعليمات وانما يتم استخدام ادوات التصميم مثل مصمم النماذج وذلك لانه يناسب اغلب الاغراض كما انه سهل الاستخدام ويتميز بالانتاجية العالية.
لكن المبرمج المحترف لابد ان تتوفر لديه مهارة برمجة واجهة البرنامج من خلال الاكواد لان ذلك سيكون اسهل فى بعض الحالات كما انه سيكون ضرورى فى حالات اخرى
تحتوى فيجوال فوكس برو على فصائل اساسية تتيح توفير امكانية برمجة واجهة البرنامج من خلال الاكواد وتعلم تلك المهارة ليس صعبا.

١ – انشاء نموذج form -:

```
myform = CREATEOBJECT("form")
myform.Show()
```

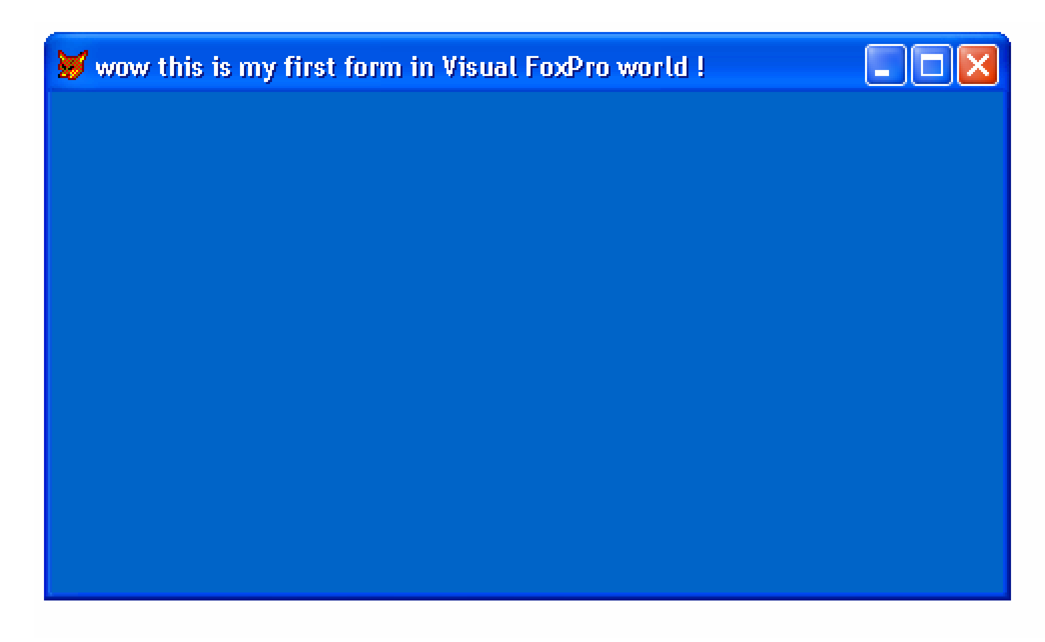
يمكنك بسهولة انشاء نموذج جديد من الفصيلة form كما فى المثال السابق وتقوم بعرضه من خلال ال method التى تسمى show()



شكل ٤١ : نتيجة تنفيذ التعليمات

وبالتاكيد يمكننا تحديد الخصائص التى نريدها للنموذج بسهولة .
فمثلا لاختيار عنوان محدد لاسم النموذج وتحديد لون خاص للخلفية:

```
Myform.caption = "wow this is my first form in Visual FoxPro world ! "
myform.backcolor = rgb(0,100,200)
myform.width = myform.width + 100
```



شكل ٤٢ : نتيجة تنفيذ التعليمات

٢ – انشاء نماذج تشمل عناصر :-

من الموكد ان النافذة او النموذج بمفرده ليس الهذف وانما نود اضافة عناصر اخرى بداخلها والمثال التالى يوضح ذلك

```
* program6.prg
form1 = createobject("myform")
form1.show()
READ events

DEFINE CLASS myform as Form
ADD OBJECT mylabel1 as label
ADD OBJECT mybtn as commandbutton
ADD OBJECT mybtn2 as commandbutton
Caption = "yes or no message"
height = 100
width = 250
mylabel1.caption = "this is my message ?"
mylabel1.top = 15
mylabel1.left = 10
mylabel1.width = 200
mybtn.caption = "yes"
mybtn.top = 50
mybtn.width = 100
mybtn.height = 30
mybtn.left = 20
mybtn2.caption = "no"
mybtn2.top = 50
mybtn2.left = 120
mybtn2.width = 100
mybtn2.height = 30
PROCEDURE mybtn.click()
thisform.Release
RETURN
ENDDEFINE
```

ونستفيد من هذا الاجراء كم جيد من المعلومات :-

١ – اولا يمكن لملف الاجراء ان يحتوى على تعليمات تنشى كائنات بجانب الفاصائل التى تحتاجها مباشرة
٢ – الامر read events يعطى التحكم للنموذج
٣ – تم انشاء فصيلة جديدة من فصيلة لم نقم نحن بكتابتها وانما هى متوفرة داخل اللغة مباشرة وهذه الفصيلة هى Form والفصيلة التى قمنا بانشائها هى Myform
٤ – تم اضافة عناصر للنموذج من خلال الجملة ADD OBJECT وتم فيها تحديد انواع هذه العناصر بعد كلمة AS
٥ – يتم التحكم بخصائص النموذج مباشرة عن طريق اسماء المتغيرات لاننا داخل فصيلة هذا النموذج ولكن سمات العناصر مثل زر الامر تشترط تحديد اسم كائن زر الامر اولا
٦ – يتم كتابة الاكواد التى تستجيب للاحداث مثل حدث click لزر الامر (يحدث عندما يضغط المستخدم بزر الفارة الايسر على زر الامر ويرفع يده ومؤشر الفارة مازال فى المساحة الخاصة بزر الامر على النموذج)
ومثال على ذلك الاجراء ()mybtn.click
٧ – ال method التى تسمى release الخاصة بالنموذج تقوم بمسحة وهى شبيه بعمل click على زر close (علامة x) فى شريط العنوان الخاص بالنموذج

ونتيجة تنفيذ هذا البرنامج كالتالى :



شكل ٤٣ : نتيجة تنفيذ البرنامج

والان يمكنك كما تعلمنا من خلال التعليمات وشاشات المساعدة التعمق فى تفاصيل برمجة الكائنات وبرمجة واجهة البرنامج من خلال التعليمات.

ملحوظة : تشمل اللغة على دوال تسهل العديد من العمليات مثل الدالة ()messagebox والتى تعرض رسالة للمستخدم والدالة ()inputbox والتى تعطى مربع نص للادخال.

```
Messagebox("every thing is done !",0,"wow")
```



شكل ٤٤ : نتيجة تنفيذ الامر

سادسا : اعداد البرامج باستخدام اسلوب الصفوف الثلاث :

يعنى ذلك تقسيم مكونات البرامج الى ٣ اجزاء جزء خاص بالبيانات وجزء خاص بواجهة البرنامج وجزء خاص بمنطق البرنامج او الادارة والتحكم – وجزء البيانات يشمل تصميم قاعدة البيانات وانشائها وينبغى فهم كيفية التعامل معها وجزء واجهة البرنامج – يعنى اعطاء واجهة للمستخدم تستقبل منه التعليمات ثم من خلال الادارة والتحكم يتم تنظيم كيفية الربط بين قاعدة البيانات و واجهة المستخدم

لقد تعلمنا كيفية ادارة البيانات وتعلمنا برمجة الكائنات والتى سوف تكون العامل الموثر فى جزء الادراة والتحكم وينبغى معرفة الان كيفية اعداد واجهة البرنامج – ثم الربط بينهما فى ملفات المشاريع .

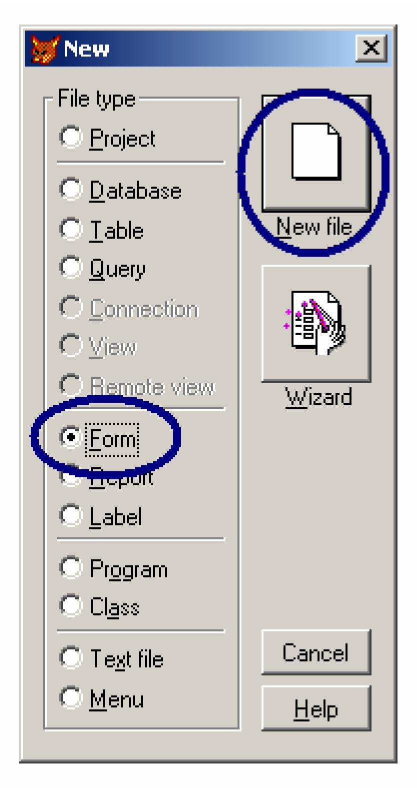


الباب الرابع

ملفات النماذج

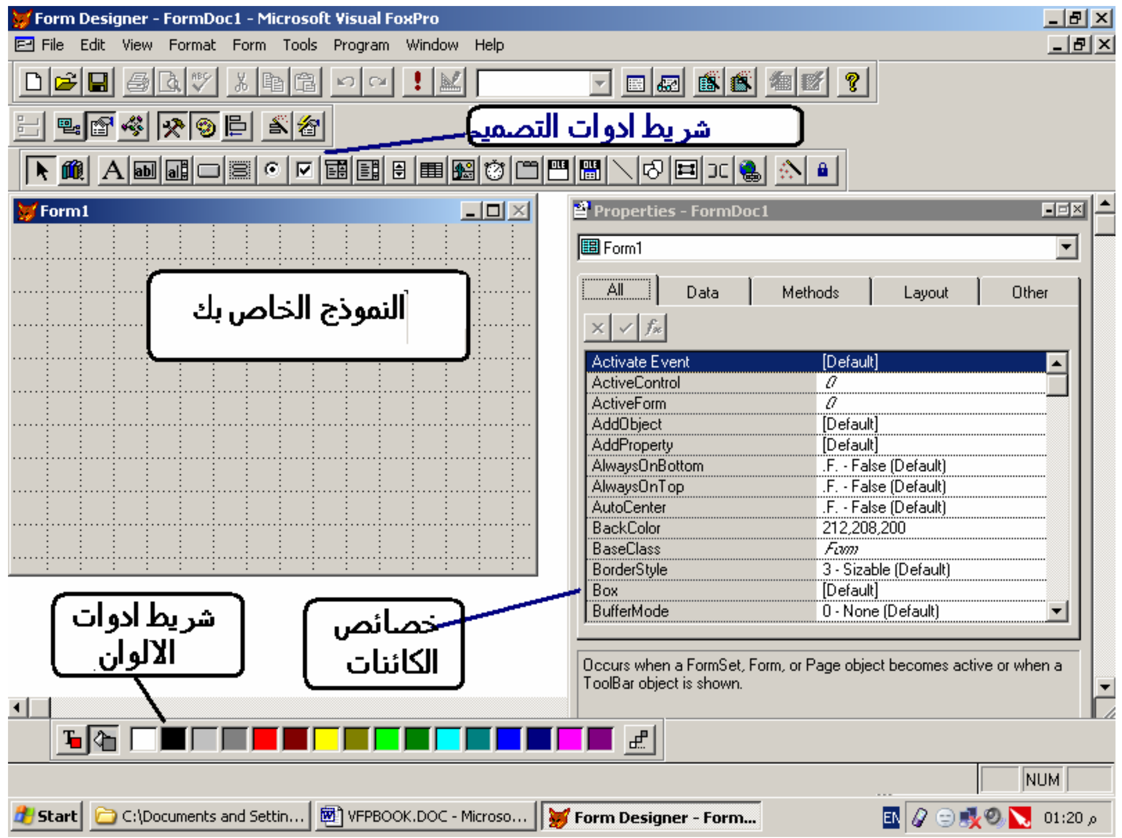
١ – انشاء نموذج جديد creating new form :-

اختار file ثم new ثم form ثم New File



شكل ٤٥: انشاء نموذج جديد

ومصمم النماذج داخل اللغة كما بالشكل التالى :



شكل ٤٦ : مصمم النماذج داخل فيجوال فوكس برو

اولا : شريط الادوات :-
يستخدم فى اضافة عناصر (كائنات جديدة) الى النموذج

ثانيا : شريط الالوان :-
يستخدم فى تلوين الكائن بسرعة (لون الكتابة – لون الخلفية)

ثالثا : نافذة الخصائص :-
تستخدم فى ضبط خصائص النموذج والعناصر التى يشملها – اغلب الخصائص مفهومة المعنى مباشرة ويمكن التعامل معها بسهولة

رابعا : نافذة النموذج :-
تعطى صورة حية مباشرة للنموذج الذى تقوم بتصميمه بحيث يمكنك اضافة العناصر اللازمة من خلال شريط الادوات والتحكم فى خصائصها من خلال نافذة الخصائص ثم بعد ذلك تكتب الاكواد الخاصة بالعناصر من خلال نافذة الاحداث.

خامسا : نافذة الاحداث :-
تظهر بمجرد الضغط بزر الفارة الايسر مرتين متتابعتين على النموذج او احد العناصر التى يحتويها

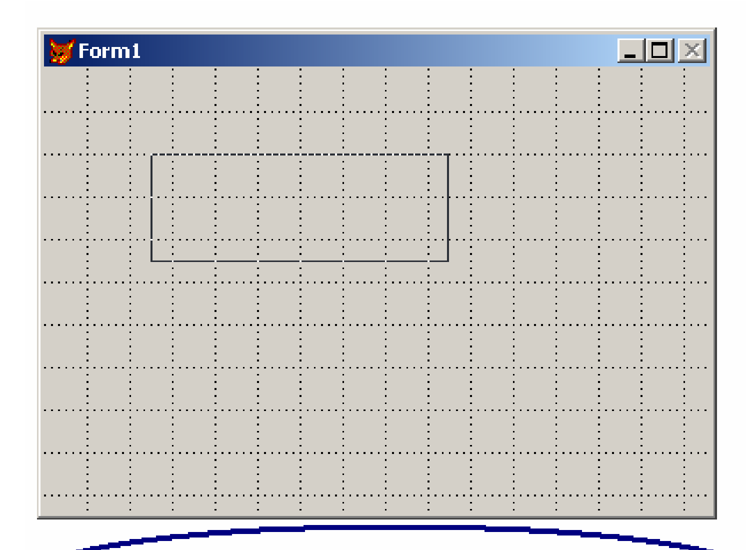
- **عناصر شريط الادوات :-**

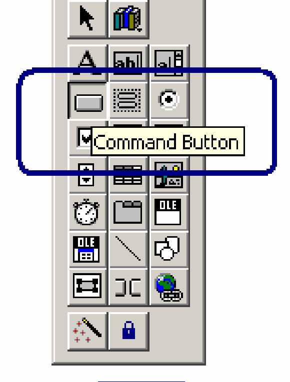
عنوان	تحديد العناصر
مربع نص	المزيد المنقص
صندوق قائمة	الجدول
زر امر	صورة
مجموعة ازرار اوامر	الموقت
صندوق خيارات	الصفحات
مربع فحص	اضافة عناصر تحكم [حاوى]
قائمة السفلية	اضافة عناصر تحكم [محدودة]
مربع قائمة	خط مائل
اشكال هندسية	الحاوى
الفاصل بين العناصر	مشير المواقع
غالق معالج البناء	غالق الازرار
المكتبات	

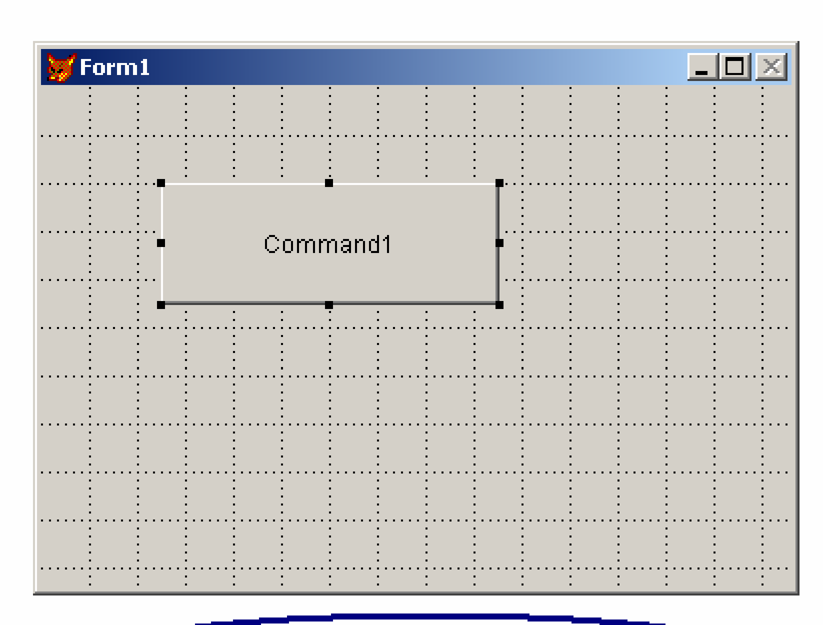
شكل ٤٧ : عناصر شريط الادوات

اضافة عنصر الى النموذج :-

يتم ذلك باختيار نوع العنصر من شريط الادوات ثم الذهاب الى النموذج ونضغط بزر الفارة الايسر ونستمر فى الضغط حتى نحدد مساحة العنصر ثم نرفع يدنا – فنجد ان العنصر قد ظهر على النموذج وتم تحديده ايضا بحيث يمكن تعديل خواصه مباشرة من خلال نافذة الخصائص



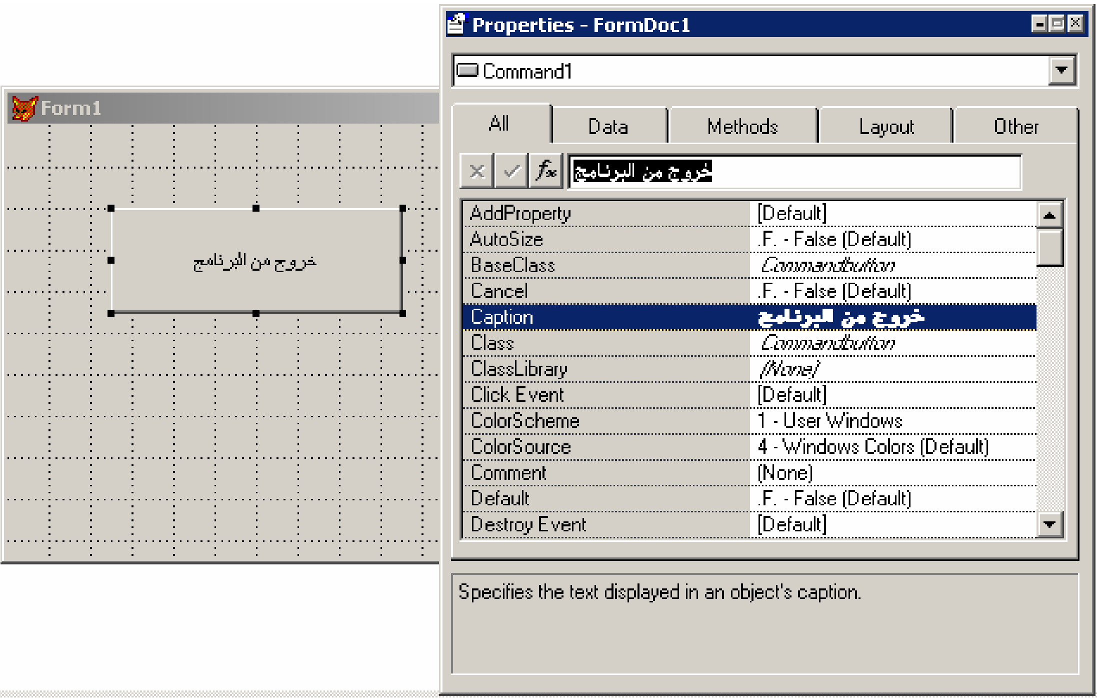




شكل ٤٨ : اضافة عنصر للنموذج

تعديل خصائص العنصر :-

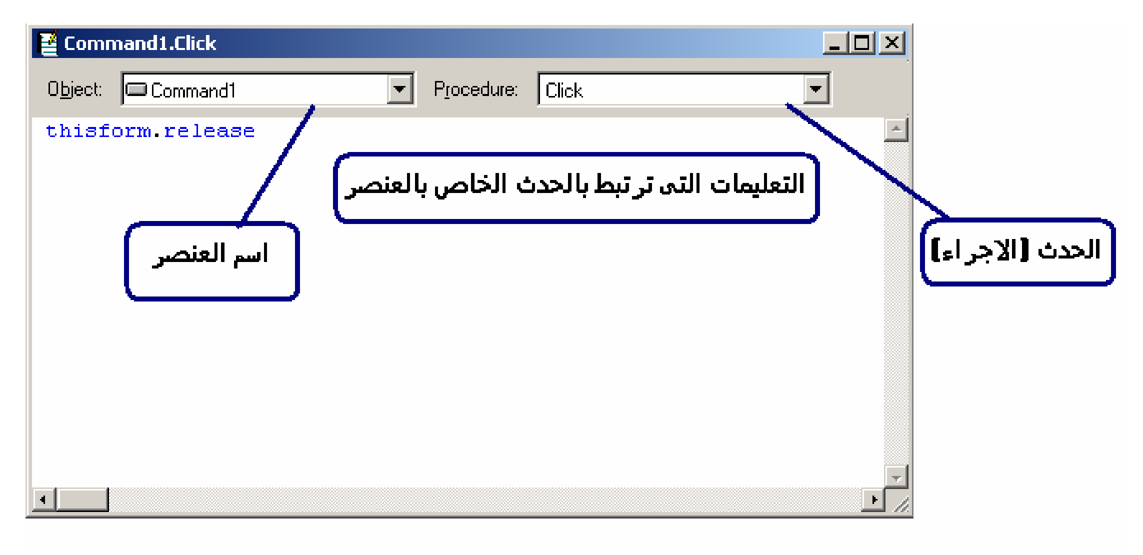
كما سبق وان ذكرنا يتم ذلك من نافذة الخصائص
فمثلا لتعديل العنوان الذى يحمله زر الامر يتم ذلك من خلال الخاصية caption



شكل ٤٩ : تعديل خواص العناصر

اضافة التعليمات الى العنصر :-

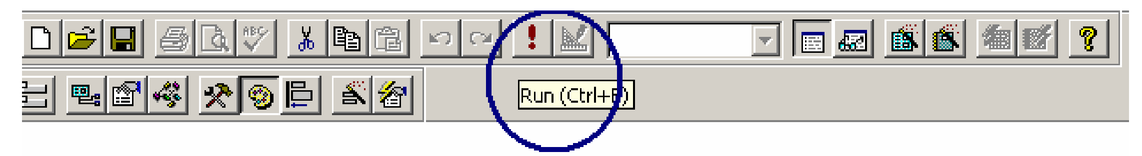
يتم ذلك من خلال نافذة الاحداث والتى نحصل عليها بالضغط مرتين على العنصر



شكل ٥٠ : اضافة تعليمات ترتبط بحدث خاص بالعنصر

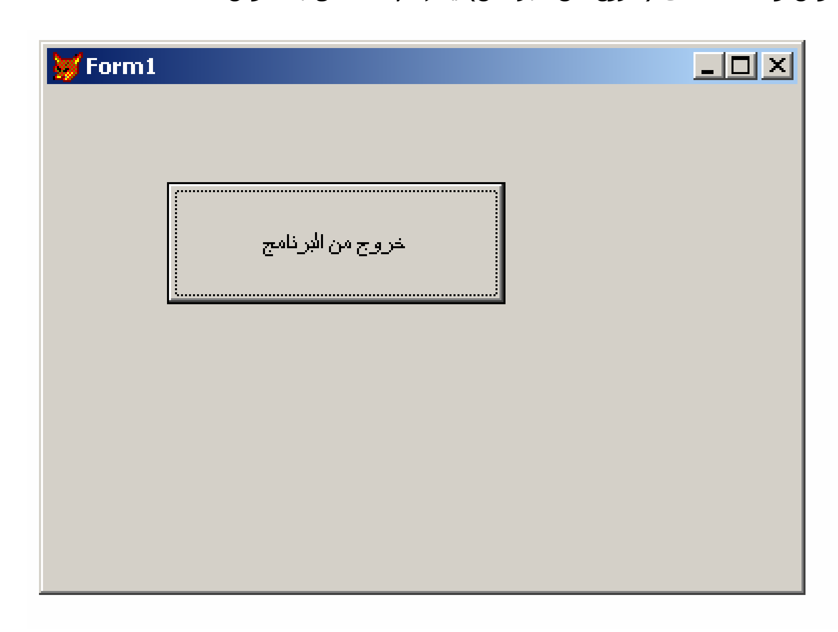
تشغيل واختبار النموذج :-

يتم ذلك من خلال الضغط على Ctrl+E وقد تظهر شاشة تطلب حفظ النموذج فى ملف اذا لم يكن قد تم ذلك من قبل – او يمكن تشغيل النموذج من خلال قائمة form ثم run او من خلال شريط الادوات



شكل ٥١ : تشغيل النموذج للاختبار

وعند تشغيل النموذج والضغط على (خروج من البرنامج) يتم انهاء العمل بالنموذج



شكل ٥٢: النموذج اثناء العمل

اهم خصائص زر الامر :

Caption	العنوان الذى يحمله زر الامر (حقل حرفى)
Forecolor	لون الكتابة (حقل رقمى)
Fontname	اسم الخط (حقل حرفى)
Fontsize	حجم الخط (حقل رقمى)
Autosize	التحجيم التلقائى (حقل منطقى)
Visible	الظهور على النموذج (نوع الحقل منطقى)
Enabled	التمكين (حقل منطقى)
Top	رقم الصف لبداية العنصر فى النموذج (حقل رقمى)
Left	رقم العمود لبداية العنصر فى النموذج (حقل رقمى)
Width	عرض العنصر (حقل رقمى)
Height	طول العنصر (حقل رقمى)
Name	اسم العنصر
ToolTipText	رسالة المعلومة السريعة

ومعظم هذه الخصائص شائعة بين العناصر ويمكن ضبطها من خلال نافذة الخصائص او من خلال التعليمات داخل الاحداث الخاصة بالنموذج او العناصر كالتالى

امثلة على ضبط الخصائص من خلال التعليمات :-

```
Thisform.command1.caption = "hello"
Thisform.command1.top = 20
Thisform.command1.enabled = .T.
Thisform.command1.forecolor = RGB(50,50,50)
```

اهم خصائص مربع النص :

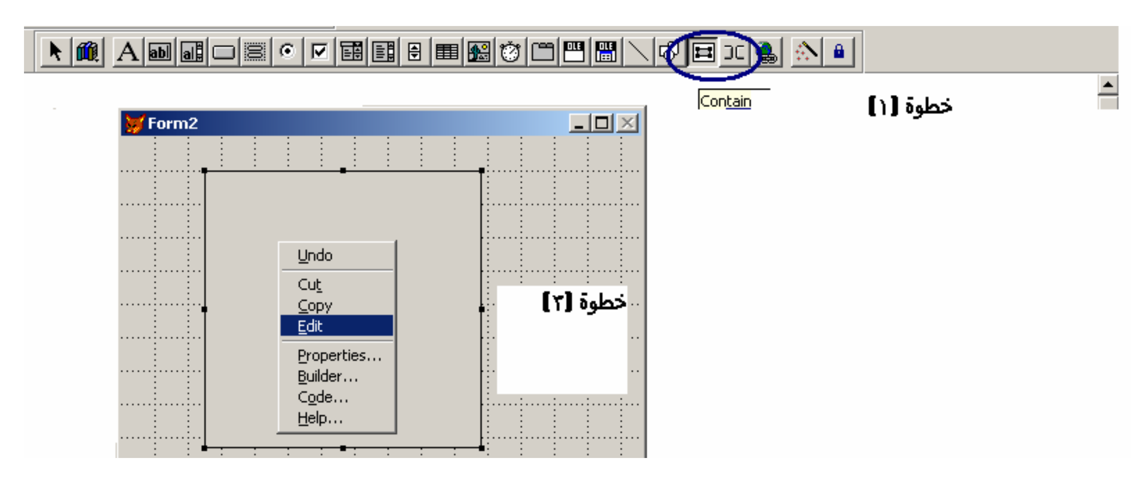
Value	القيمة التى يحتويها مربع النص
Forecolor	لون الكتابة (حقل رقمى)
Fontname	اسم الخط (حقل حرفى)
Fontsize	حجم الخط (حقل رقمى)
Autosize	التحجيم التلقائى (حقل منطقى)
Visible	الظهور على النموذج (نوع الحقل منطقى)
Enabled	التمكين (حقل منطقى)
Top	رقم الصف لبداية العنصر فى النموذج (حقل رقمى)
Left	رقم العمود لبداية العنصر فى النموذج (حقل رقمى)
Width	عرض العنصر (حقل رقمى)
Height	طول العنصر (حقل رقمى)
Name	اسم العنصر
backcolor	لون الخلفية
RightToLeft	كتابة من اليمين لليسار (حقل منطقى)
Alignment	المحاذاة (حقل رقمى)
MaxLength	اقصى طول للادخال
InputMask	شروط الادخال (مثلا ٩٩٩٩ تعنى يقبل ارقام فقط واقصى طول اربعة)
ToolTipText	رسالة المعلومة السريعة (حرفى)

ويمكنك من خلال (شاشات المساعدة الخاصة باللغة – التجربة) معرفة كيفية التعامل مع جميع العناصر التى قد يشتمل عليها النموذج وتاكد انك لن تجد صعوبة فى ذلك وانما فقط تحتاج الى بعض الوقت.

العناصر الحاوية :-

هى العناصر التى يمكن ان تشمل عناصر اخرى بداخلها شانها شان النموذج فى ذلك ومن امثلتها & container pageframe اى عنصر الحاوى وعنصر الصفحات

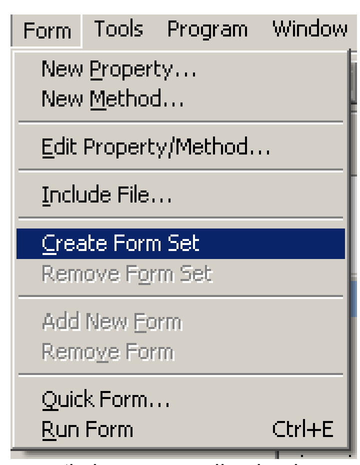
وعند اضافة مثل هذه العناصر للنموذج وتود تعديل محتوياتها باضافة العناصر اليها والتحكم بها – يتم ذلك بتحديد العنصر اولا ثم ضغط الزر الايمن للفارة ثم اختيار edit لكى تحرر محتوياتها



احتواء ملف النماذج على اكثر من نموذج :

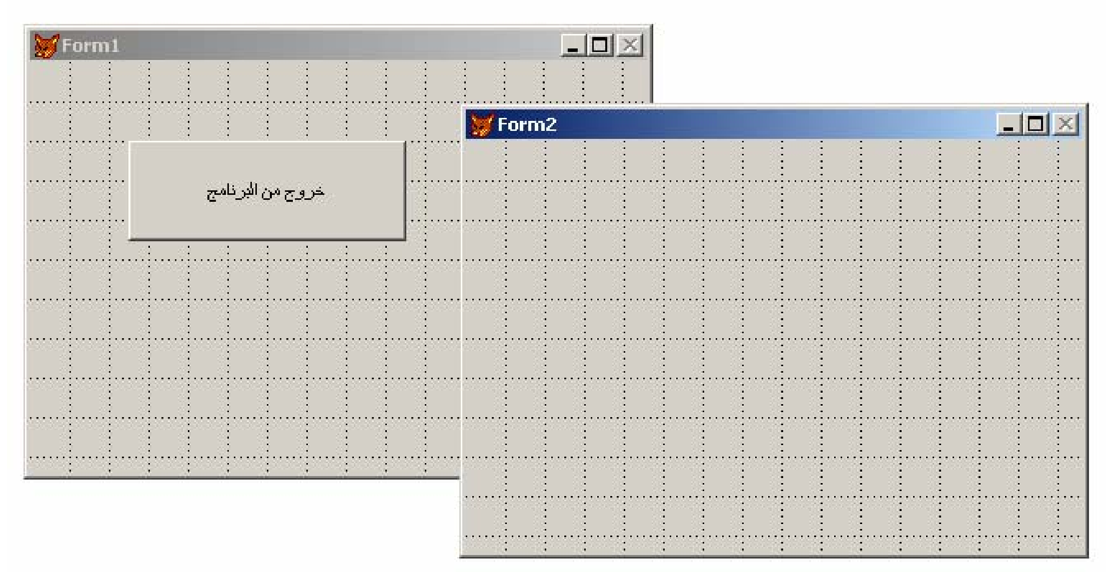
نعم يمكن ان يحتوى ملف النماذج على اكثر من نموذج ونقوم بعمل ذلك اذا كان هناك ارتباط شديد جدا بين هذه النماذج – ولكن من الافضل دائما ان يكون كل نموذج فى ملف مستقل مما يسهل بعد ذلك نقل النماذج بين المشاريع وغيرها.

من قائمة form اختار create form set



شكل ٥٣: جعل ملف النموذج يشمل اكثر من نموذج

والان لاضافة نموذج جديد اختار من قائمة form الاختيار Add new form ولازالته يمكن من خلال remove form



شكل ٥٤: اكثر من نموذج فى ملف واحد

ولكى تحدد التعامل مع نموذج معين من اى نموذج فى الملف يتم ذلك كالتالى

thisformset.formname.property/event

مثال :

example : thisformset.form2.release

استدعاء ملف النماذج :

يمكنك من داخل النموذج او من اى مكان يقبل التعليمات استدعاء ملف نموذج من خلال الامر Do Form والذى ينبغى ان تكون قد لاحظته اذا كنت تديم النظر الى نافذة الاوامر

مثال :

```
Do form myotherform.scx
```

ونلاحظ ان ملف النموذج ياخذ الامتداد scx. وكتابة الامتداد فى الامر غير ضرورى

```
Do form myotherform
```

ولا يشترط كتابة المسار بفرض ان المجلد الحالى يوجد به ملف النموذج.

الخصائص الهامة للنموذج :

فى الواقع ان اغلب خصائص النموذج فى غاية الاهمية ولكن هناك خصائص محددة فى غاية الحيوية وهى الخصائص التى تحدد فى اى بيئة سوف يعمل النموذج

١ – هل سيعمل فى بيئة فيجوال فوكس برو
٢ – هل سيعمل كبرنامج مستقل بعيدا عن واجهة فيجوال فوكس برو

البعض قد يسال الان
س : ما هى حاجتى لان يعمل برنامجى داخل بيئة فيجوال فوكس برو ؟
جـ : قد يكون برنامجك الذى تطوره عبارة عن برنامج مساعد لتطوير البرامج داخل اللغة وعندها فانك تود ان يتفاعل البرنامج مع واجهة اللغة.
كما ان التوافق مع البرمجيات التى تم تطويرها باستخدام فوكس برو قبل اصدار فيجوال فوكس برو يتطلب ذلك حيث كانت فوكس برو لاتصدر برامج مستقلة وانما تشترط وجود قاعدة البيانات فوكس برو لتشغيل البرنامج – لكن فيجوال فوكس برو لغة برمجة تنتج برامج مستقلة
يمكن اثناء تطوير البرامج القديمة التى تعمل فى بيئة فوكس برو باستخدام فيجوال فوكس برو ان تجعلها برامج مستقلة بدون جهد يذكر او اى تعديل حيث عند انشاء برنامج مستقل من خلال المشاريع تعطيك اللغة داخل الملف التنفيذى exe. الذى تنشئه واجهة مشابهة لفيجوال فوكس برو بمعنى (نافذة رئيسية وشريط قوائط وشريط ادوات وغيرها)

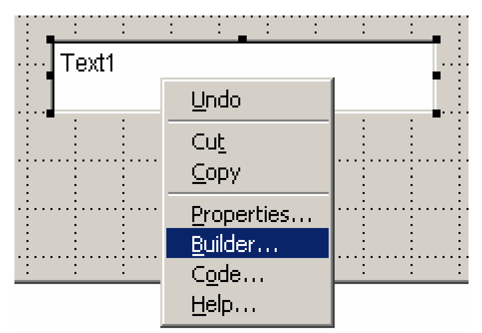
س : لقد اختلط الامر على – ماذا افعل بالتحديد عند تطوير برامجى ؟
جـ : كل ما عليك ان تحدد اسلوب لكتابة برنامجك الذى سوف يعمل بصفة مستقلة عن فيجوال فوكس برو
الاسلوب الاول : تكتب البرنامج كانه سيعمل من داخل فيجوال فوكس برو وعند الترجمة لكى يعمل برنامجك مستقلا تحصل على شاشة شبيهة بواجهة فيجوال فوكس برو لكى يعمل منها برنامجك ويمكنك التحكم فى تلك الشاشة بمعنى اختيار اسم للنافذة الرئيسية واختار قائمة خاصة بك

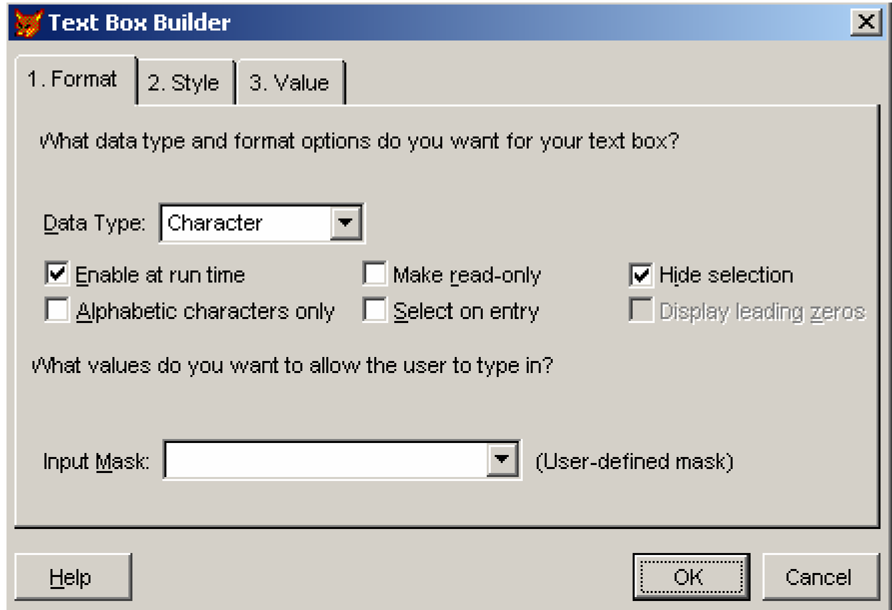
الاسلوب الثانى : تصمم برنامجك ليعمل مستقلا من البداية عن واجهة فيجوال فوكس برو وتقوم بعمل واجهتك الخاصة من البداية وعند الترجمة تلغى واجهة فيجوال فوكس برو تماما (يتم ذلك بسطر واحد من التعليمات) وتنادى واجهتك للعمل ويفضل ذلك الاسلوب لانه الاسلوب الجديد ويعطى مرونة اكثر من حيث كون البرنامج يعمل فى ملىء الشاشة وغيرها من الامكانيات (خاصة اذا كان البرنامج الذى تطوره عبارة عن مجموعة برامج متكاملة مجتمعة معا وتريدها ان تعمل على التوازى وفى نفس التطبيق)

Showwindow	تعطى ٣ اختيارات 0 – in screen (default) تعرض داخل واجهة فيجوال فوكس برو 1 – in top level form تعرض داخل نموذج مستقل 2 – as top level form يعرض كنموذج مستقل بديل لواجهة فيجوال فوكس برو
desktop	حقل منطقى يحدد هل يطفو النموذج على سطح المكتب ام لا
titlebar	هل يظهر شريط العنوان ام لا
controlbox	هل يظهر صندوق التحكم ام لا
borderstyle	التحكم فى حدود النوذج
Autocenter	هل يظهر النموذج تلقائيا فى منتصف الشاشة
icon	ايقونة النموذج
picture	صورة تعرض كخلفية للنموذج ويمكن ان تكون صورة صغيرة لخامة مثلا (دهب – فضة) ويتم تكرارها لملىء النموذج

معالج البناء الخاص بعناصر النموذج :-

نعم هناك معالج خاص بكل عنصر فى النموذج تقريبا – وهذا المعالج يساعد على ضبط الخصائص الاساسية للعنصر فمثلا معالج مربع النص يسهل عملية ربطه بقاعدة البيانات وهكذا.
ولتشغيل معالج البناء حدد العنصر ثم اضغط الزر الايمن لتظهر القائمة القصيرة فاختار منها builder

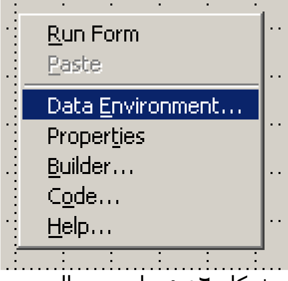




شكل ٥٥ : بناء مربع النص

بيئة البيانات داخل النموذج :-

وهى خاصة بالجداول التى يتم فتحها مباشرة بمجرد تشغيل النموذج وهذه الجداول يمكن اضافتها او يتم اضافتها تلقائيا اذا تم ربط حقول الادخال (مربعات النص) بملفات البيانات

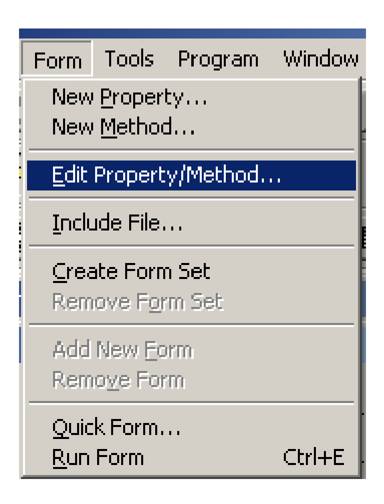


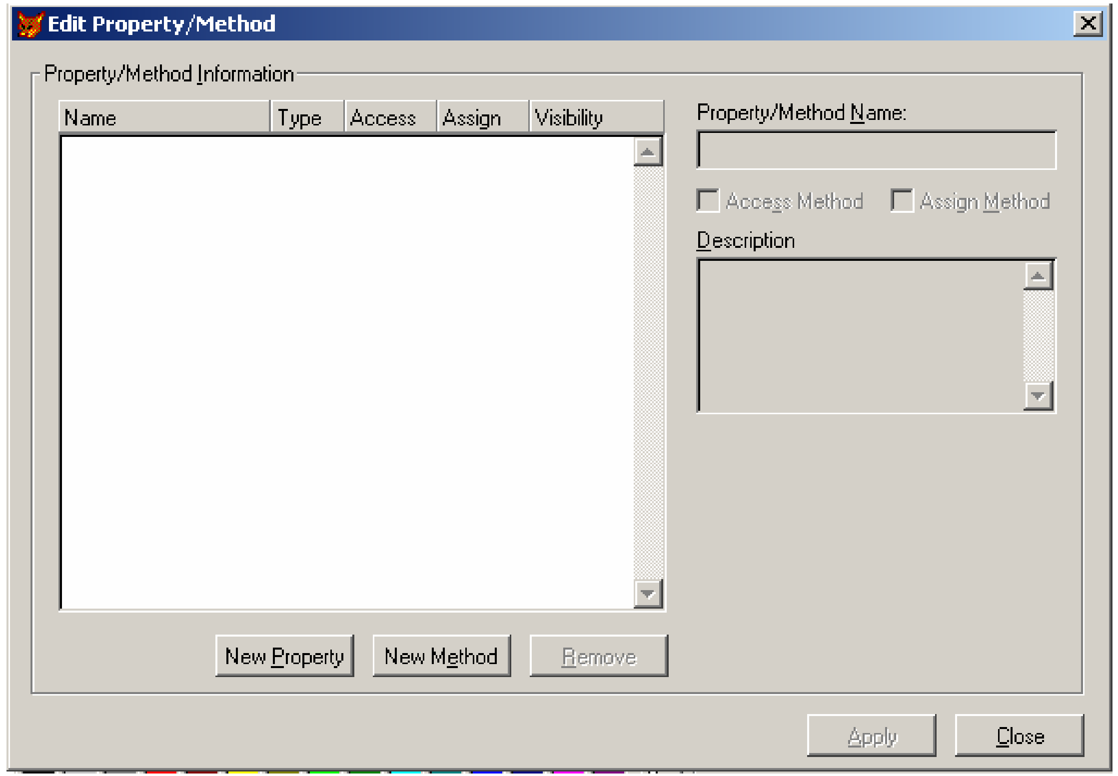
شكل ٥٦ : بناء مربع النص

ملحوظة : يمكن بدلا من ربط عناصر النموذج بملفات البيانات ان يتم برمجة ذلك يدويا من خلال فتح ملفات البيانات عند بداية تشغيل النموذج (حدث load) او حدث (init) وغلقها فى حدث release

اضافة سمات جديدة للنموذج :-

بالفعل يمكن ذلك لتسهيل الكثير من عمليات البرمجة
ويتم ذلك من خلال ٣ خيارات بقائمة form هما new property,new method & edit property/Method



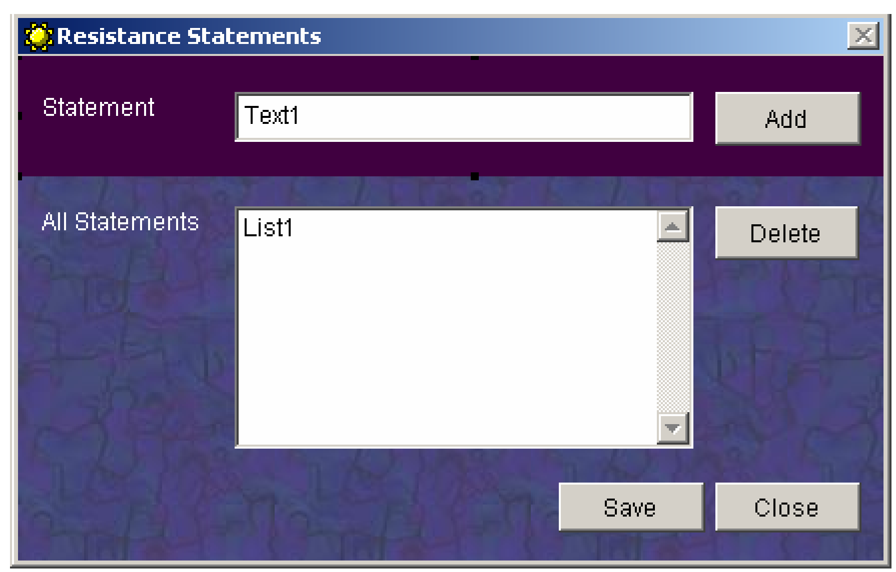


شكل ٥٧ : تعديل سمات النموذج

وبالتالى يمكن اضافة سمات جديدة والتعامل معها مباشرة من خلال التعليمات.

مثال على نموذج هادف :-

سوف ندرس الان نموذج بسيط من احد المشاريع الحية (مشروع قمت بعمله منذ فترة) والهدف من ذلك المثال معرفة كيفية التعامل مع مربع القائمة (listbox) باضافة عناصر اليها والتعرف على محتوياتها وكذلك كيف ان هذا النموذج يتعامل مع ملف البيانات



شكل ٥٨: مثال جيد لمربع القائمة

هذا النموذج يشمل عنوانين هما statement و All statements
ومربع نص ومربع قائمة واربعة من ازرار الاوامر هما add,delete,save & close

```
* event : form1_init()        الحدث الخاص بالنموذج هو
select 25
if .not. reccount() = 0
for x = 1 to reccount()
goto x
ThisForm.List1.additem(statement)
next
endif
```

ه١٥ التعليمات فى بداية النموذج تحدد ملف البيانات الموجود فى المنطقة ٢٥ والذى سبق فتحه من قبل برنامج رئيسى سابق باستخدام الامر use ومن هنا نجد انه عند فتح ملفات البيانات التى نتعامل معها مرة واحد كل ملف بالتاكيد فى منطقة خاصة به يمكن تحديد الملف فى اى نموذج بسهولة باستخدام الامر select

الحقل statement موجود داخل ملف البيانات
الاجراء additem يستخدم لاضافة عنصر الى القائمة

```
* addbutton_click()  حدث الضغط على زر الاضافة
If .not. empty(alltrim(ThisForm.Text1.value))
THISFORM.LockScreen = .T.
            nCnt = 1
            DO WHILE nCnt <= THISform.list1.ListCount
            IF upper(alltrim(THISform.list1.listitem(nCnt))) ==;
      upper(alltrim(ThisForm.Text1.value))
       mymsg = messagebox("The statement is already added",6,"Sorry")
            return
            ELSE
                  nCnt = nCnt + 1
            ENDIF
            ENDDO
            THISFORM.LockScreen = .F.
ThisForm.List1.additem(alltrim(ThisForm.Text1.value))
Else
mymsg = messagebox("enter the statement ",6,"sorry")
Endif
```

من هذه التعليمات نجد ان الخاصية Listcount تعطى عدد العناصر فى القائمة
الخاصية listitem تاخذ رقم العنصر كمعطى وتعطى قيمته النصية

اجمالى هذه التعليمات بان يتم اضافة الجملة التى تتم كتابتها الى القائمة بعد التاكد من عدم وجودها من قبل

والتعليمات فى زر الحذف كالتالى

```
* deletebtn_click()  حدث الحذف
THISFORM.LockScreen = .T.
            nCnt = 1
            DO WHILE nCnt <= THISform.list1.ListCount
                  IF THISform.list1.Selected(nCnt)
                  THISform.list1.RemoveItem(nCnt)
                  ELSE
                        nCnt = nCnt + 1
                  ENDIF
            ENDDO
THISFORM.LockScreen = .F.
```

وفى هذه التعليمات نجد الاجراء selected الذى ياخذ رقم العنصر فى القائمة ويحدد هل هو محدد ام لا
والاجراء removeitem يستخدم لازالة عنصر من القائمة عن طريق رقمه

والتعليمات المكتوبة فى زر الحفظ كالتالى

```
* savebtn_click() event  حدث زر الحفظ
select 25
delete all
pack
goto top
if .not. ThisForm.List1.listcount = 0
for x = 1 to ThisForm.List1.listcount
append blank
replace statement with ThisForm.List1.listitem(X)
next
endif
```

وهذه التعليمات تنقل بيانات القائمة الى ملف البيانات

AddItem(parameter_string)	اضافة عنصر الى القائمة
Listcount	عدد عناصر القائمة
ListItem(parameter_Numeric)	قيمة العنصر داخل القائمة
Selected(parameter_Numeric)	هل العنصر محدد ؟
RemoveItem(parameter_Numberic)	ازالة عنصر من القائمة

ملحوظة : نستخدم الخاصية Lockscreen الخاصة بالنموذج لمنع حدوث رعشة اثناء عمل البرنامج مع كثرة التغيرات فى شاشة العرض .

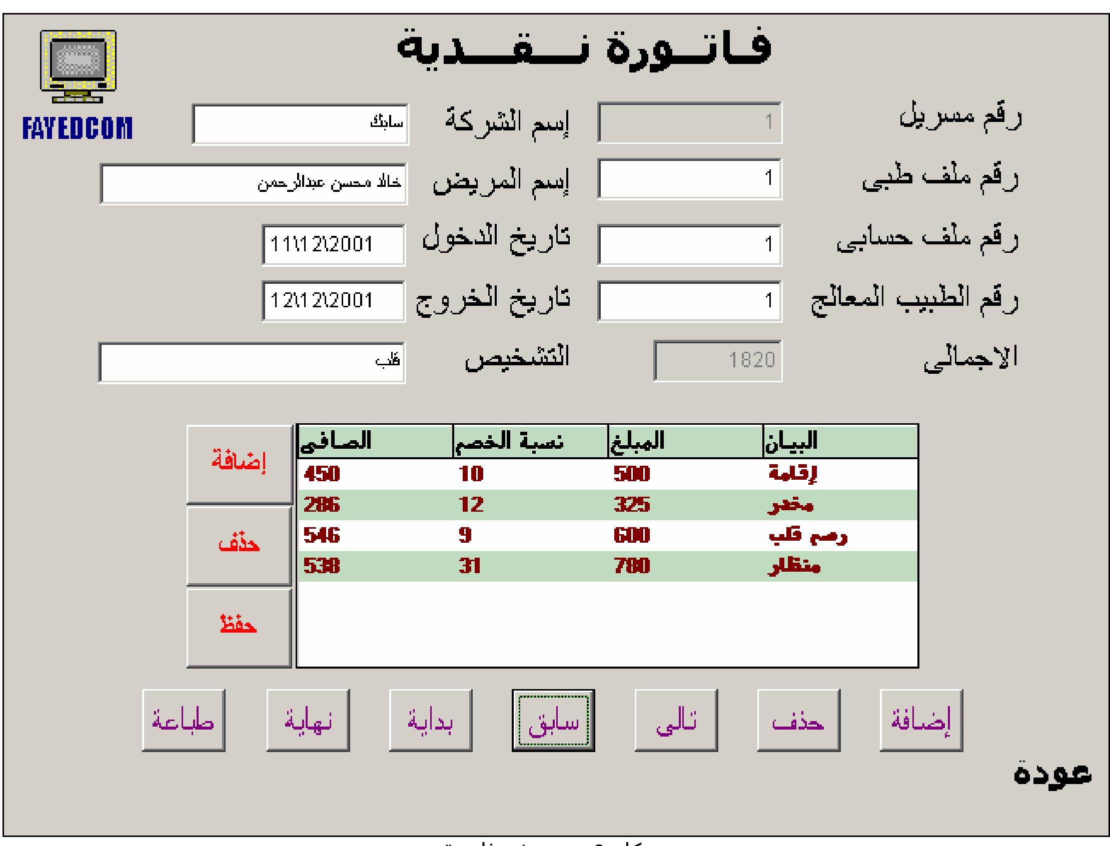
اذا كنت تود تطبيق هذا المثال كما هو قم بعمل ملف بيانات وليكن mystatements.dbf بحيث يحتوى على حقل واحد حرفى سعة ١٠٠ حرف باسم statement
وقبل تشغيل النموذج اكتب التعليمات التالية

```
Select 25
Use mystatements
```

وبذلك يعمل النموذج بصورة جيدة.

مثال على نموذج فاتورة :-

هذا المثال جيد وقوى ولكنك لن تشعر باى صعوبه به نظرا لقوة فيجوال فوكس برو .



شكل ٥٩: نموذج فاتورة

```
* form load event حدث تحميل النموذج
SET DELETE ON
Select 0
USE "fatnnum.dbf" exclusive && فتح ملف خاص برقم الفاتورة

* form unload event
close all

* add new invoice (click) event  حدث اضافة فاتورة جديدة
local myinvnum
select fatnnum
myinvnum = invnum + 1
replace invnum with invnum + 1
SELECT FATN
append blank
replace fatnum with myinvnum
thisform.refresh

* delete invoice button (Click event) حدث حذف بيانات فاتورة
delete
goto top
thisform.refresh
```

```
* next invoice button event حدث الانتقال للفاتورة التالية
SELECT FATN
IF .NOT. EOF()
skip 1
 IF EOF()
  SKIP -1
 ENDIF
ENDIF
thisform.refresh

* prev invoice button event حدث الانتقال للفاتورة السابقة
SELECT FATN
IF .NOT. BOF()
skip -1
ENDIF
thisform.refresh

* first invoice button (click event ) حدث الانتقال لاول فاتورة
SELECT FATN
if .not. reccount() = 0
goto top
endif
thisform.refresh

* last invoice button (click event) حدث الانتقال لاخر فاتورة
SELECT FATN
if .not. reccount() = 0
goto bottom
endif
thisform.refresh

* print button (click event) حدث طباعة الفاتورة
local myrec
SELECT FATN
myrec = RECNO()
SET FILTER TO RECNO() = myrec
SELECT FATNDAT
REPORT FORM "fatn.frx" FOR FATNDAT->FATNUM = myrec PREVIEW IN invform
SELECT FATN
SET FILTER TO
```

هنا تجد امرا جديدا هو report form والذى يستخدم لاستدعاء ملفات التقارير والتى يتم تصميمها بسهولة وبطريقة مشابهة لمصمم النماذج من خلال مصمم التقارير الخاص باللغة.

ملاحظة invform هو اسم النموذج الخاص بالفاتورة

```
* add item button (click event) اضافة عنصر للفاتورة
SELECT FATN
mynum = FATNUM
SELECT FATNDAT
APPEND BLANK
REPLACE FATNUM WITH mynum
ThisForm.Grid1.REFRESH

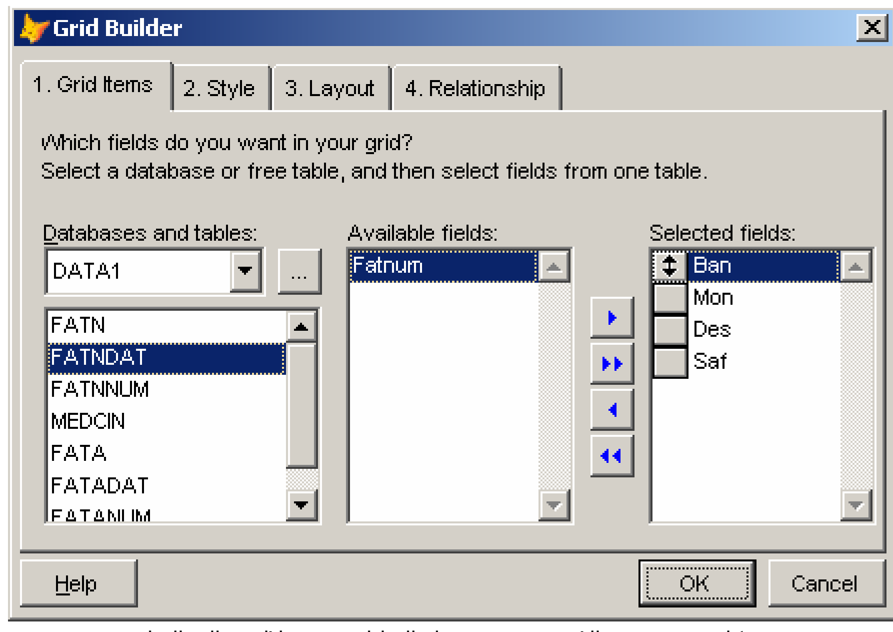
* delete item button (click event) حذف عنصر من الفاتورة
local mah,mah2
SELECT FATNDAT
DELETE
GOTO TOP
ThisForm.GRID1.REFRESH
SELECT FATN
```

```
MAH2 = FATNUM
SELECT FATNDAT
REPLACE SAF WITH MON - ( MON /100 * DES )
ThisForm.GRID1.REFRESH
MAH = 0
SET FILTER TO FATNUM = MAH2
GOTO TOP
DO WHILE .NOT. EOF()
MAH = MAH + SAF
SKIP 1
ENDDO
SET FILTER TO
SELECT FATN
REPLACE FATSUM WITH MAH
ThisForm.Text7.REFRESH

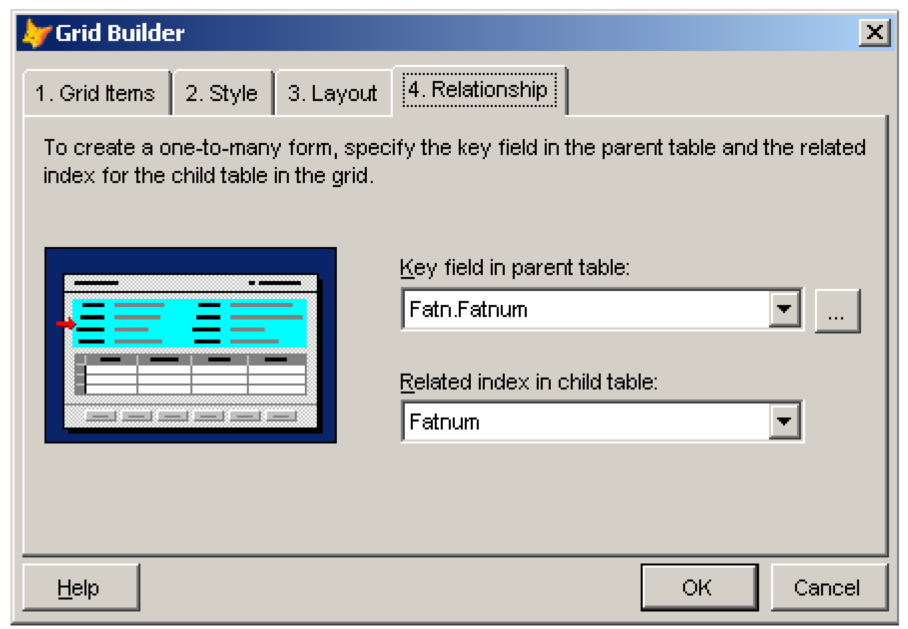
* save item button (click event) حفظ بيانات العنصر من الفاتورة
local mah,mah2
SELECT FATN
MAH2 = FATNUM
SELECT FATNDAT
REPLACE SAF WITH MON - ( MON /100 * DES )
ThisForm.GRID1.REFRESH
MAH = 0
SET FILTER TO FATNUM = MAH2
GOTO TOP
DO WHILE .NOT. EOF()
MAH = MAH + SAF
SKIP 1
ENDDO
SET FILTER TO
SELECT FATN
REPLACE FATSUM WITH MAH
ThisForm.Text7.REFRESH

* return label (click event) زر العودة
thisform.release
```

من خلال معالج البناء الخاص بالـ grid يتم تحديد بيانات جدول البيانات Data Grid وكذلك تحديد العلاقات

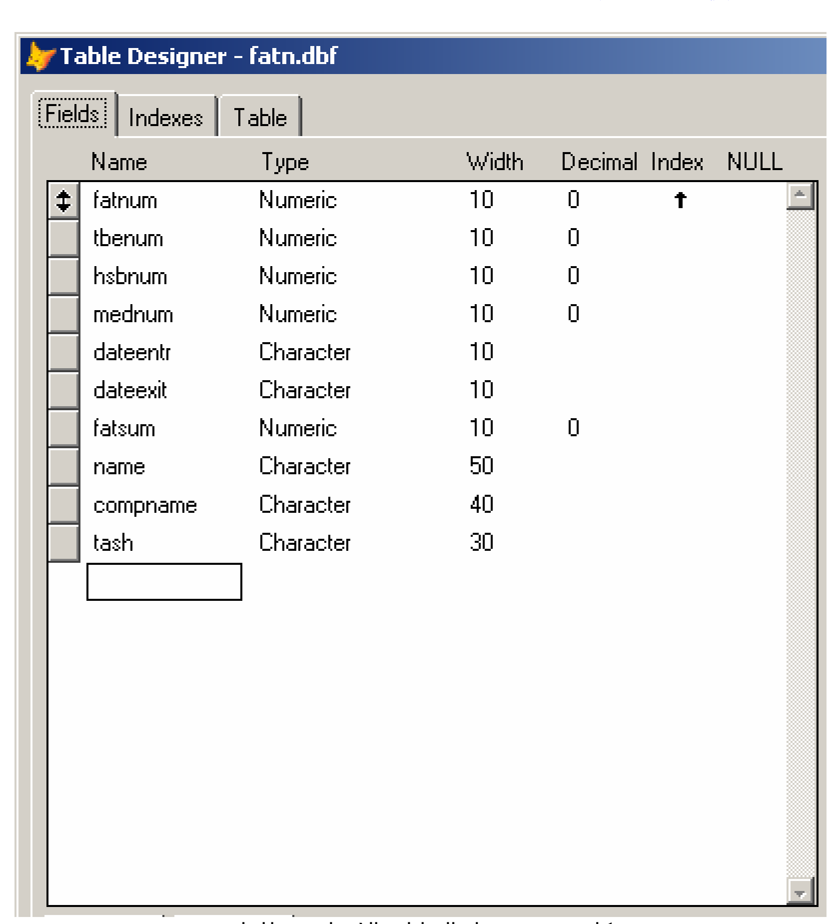


شكل ٦٠: تحديد الاعمدة فى جدول البيانات من خلال معالج البناء



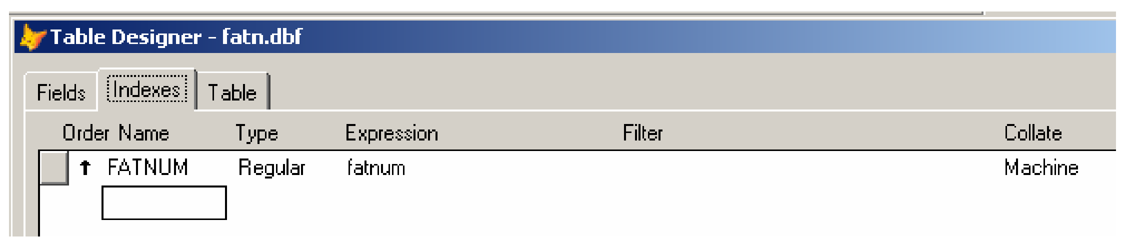
شكل ٦١ : تحديد العلاقات فى جدول البيانات من خلال معالج البناء

وتصميم قاعدة البيانات عبارة عن ٣ ملفات كالتالى :

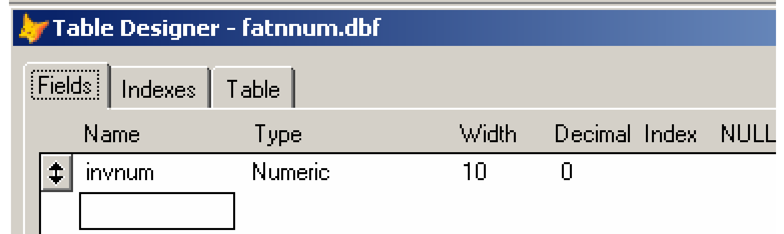


Name	Type	Width	Decimal	Index	NULL
fatnum	Numeric	10	0	↑	
tbenum	Numeric	10	0		
hsbnum	Numeric	10	0		
mednum	Numeric	10	0		
dateentr	Character	10			
dateexit	Character	10			
fatsum	Numeric	10	0		
name	Character	50			
compname	Character	40			
tash	Character	30			

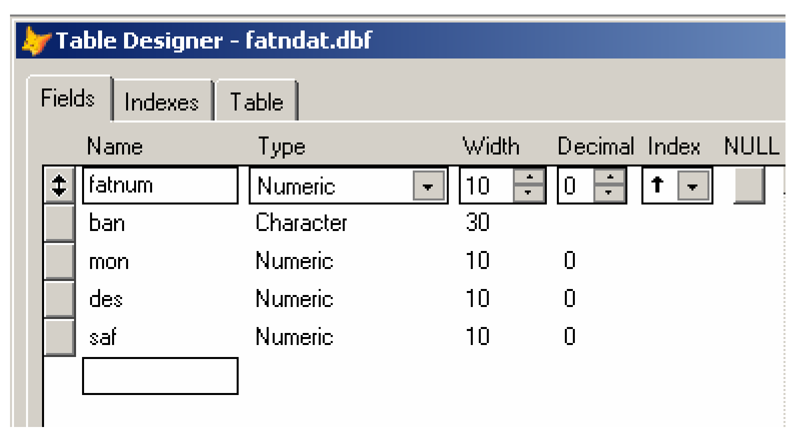
شكل ٦٢: جدول البيانات الاساسية للفاتورة



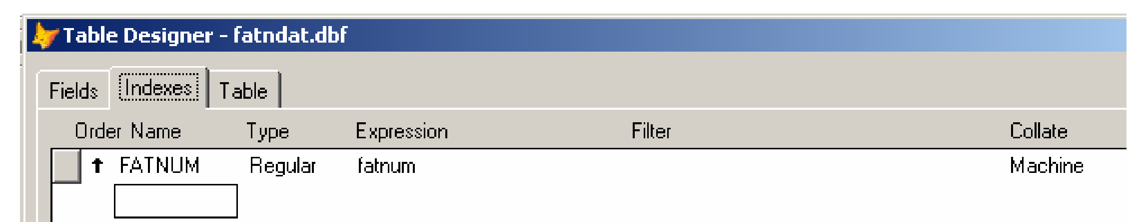
شكل ٦٢: فهرسة البيانات الاساسية للفاتورة تبعا لرقمها



شكل ٦٣: ملف خاص بتسجيل رقم الفاتورة



شكل ٦٤: ملف خاص ببيانات عناصر الفاتورة



شكل ٦٥ : فهرسة البيانات الاساسية لعناصر للفاتورة تبعا لرقمها



الباب ا امس

ملفات القوائم

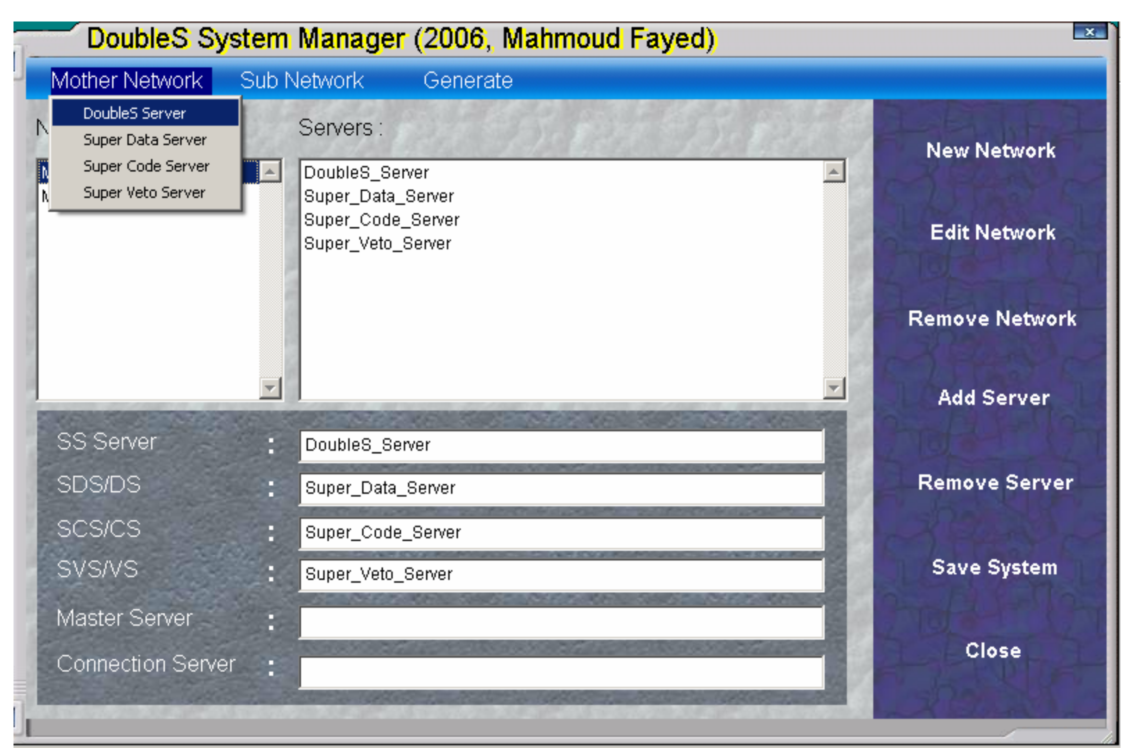
مقدمة هامة :-

شريط القائمة عنصر هام فى برنامجك ويجب اجادة كيفية برمجته – ان لغة البرمجة فيجوال بيسك ٦ تتيح امكانية اضافة شريط القائمة الى النموذج الذى تصممه مباشرة – لكن هنا فى فيجوال فوكس برو الامر يختلف حيث يتم تصميم القائمة فى ملف منفصل ثم بعد ذلك يتم استدعائها لتعمل كبديل لقائمة واجهة فيجوال فوكس برو او يتم اضافتها الى نموذجك الخاص بشرط ان يكون نوذج مستقل.

س : لماذا ملفات القوائم فى ملف منفصل ؟؟؟؟
جـ : تلك من ملامح قوة فيجوال فوكس برو – ان المعنى البسيط يقول ان وجود القائمة فى ملف منفصل يسمح بسهولة نقلها من نموذج لاخر – ولكن المعنى البعيد هو ان فيجوال فوكس برو لغة برمجة قوية تحتوى تعليمات لبرمجة شريط القائمة من الصفر ولذلك ان لا تستدعى ملف القائمة الذى تصممه مباشرة – بل تنتج ملف يحتوى على تعليمات اللغة مما يجعلك تعدل فى شريط القائمة او تاخذ اجزاء منه وتضيفها داخل النموذج على هيئة قائمة سريعة

ملحوظة : العديد من المبرمجين المحترفين لا يستخدمون ملفات القوائم مباشرة بل ياخذونها جزء جزء ويضيفونها للنموذج مما يسمح بعمل skin خاص بالبرنامج (شكل جرافك جيد).

والشكل التالى يوضح مثال لذلك وهو من احد البرامج التى سبق وان قمت بعملها.



شكل ٦٦: شكل جرافك للقوائم

وهذه هى التعليمات المسئولة عن اظهار القائمة
ملحوظة: العلامة \ فى الاختيار تعنى انه غير منشط لا يمكن اختياره Enabled false

```
DEFINE POPUP mencontex SHORTCUT RELATIVE FROM 3.5,6
IF thisform.list1.ListIndex = 1
DEFINE BAR 2 OF mencontex PROMPT  " DoubleS Server"
DEFINE BAR 5 OF mencontex PROMPT  " Super Data Server"
DEFINE BAR 7 OF mencontex PROMPT  " Super Code Server"
DEFINE BAR 9 OF mencontex PROMPT  " Super Veto Server"
ELSE
DEFINE BAR 2 OF mencontex PROMPT  "\ DoubleS Server"
DEFINE BAR 5 OF mencontex PROMPT  "\ Super Data Server"
DEFINE BAR 7 OF mencontex PROMPT  "\ Super Code Server"
DEFINE BAR 9 OF mencontex PROMPT  "\ Super Veto Server"
ENDIF
```

```
ON SELECTION BAR 2 OF mencontex _selec=1
ON SELECTION BAR 5 OF mencontex _selec=2
ON SELECTION BAR 7 OF mencontex _selec=3
ON SELECTION BAR 9 OF mencontex _selec=4
ACTIVATE POPUP mencontex
DO case
   CASE _selec = 1
   DO FORM ssser
   CASE _selec = 2
   DO FORM sdataser
   CASE _selec = 3
   DO FORM scodeser
   CASE _selec = 4
   DO FORM svetoser
ENDCASE
```

اما بالنسبة للشريط فهو عبارة عن مجموعة من العناوين تم الغاء خلفيتها من خلال الخاصية backstyle لتظهر الصورة التى خلفها ويتم فى جميع العناوين حدث تحرك الماوس كتابة الاتى

```
* Label - mousemove event
thisform.mybar.Top = this.Top
thisform.mybar.Left = this.Left
thisform.mybar.Width = this.Width + 5
thisform.mybar.Visible = .T.
```

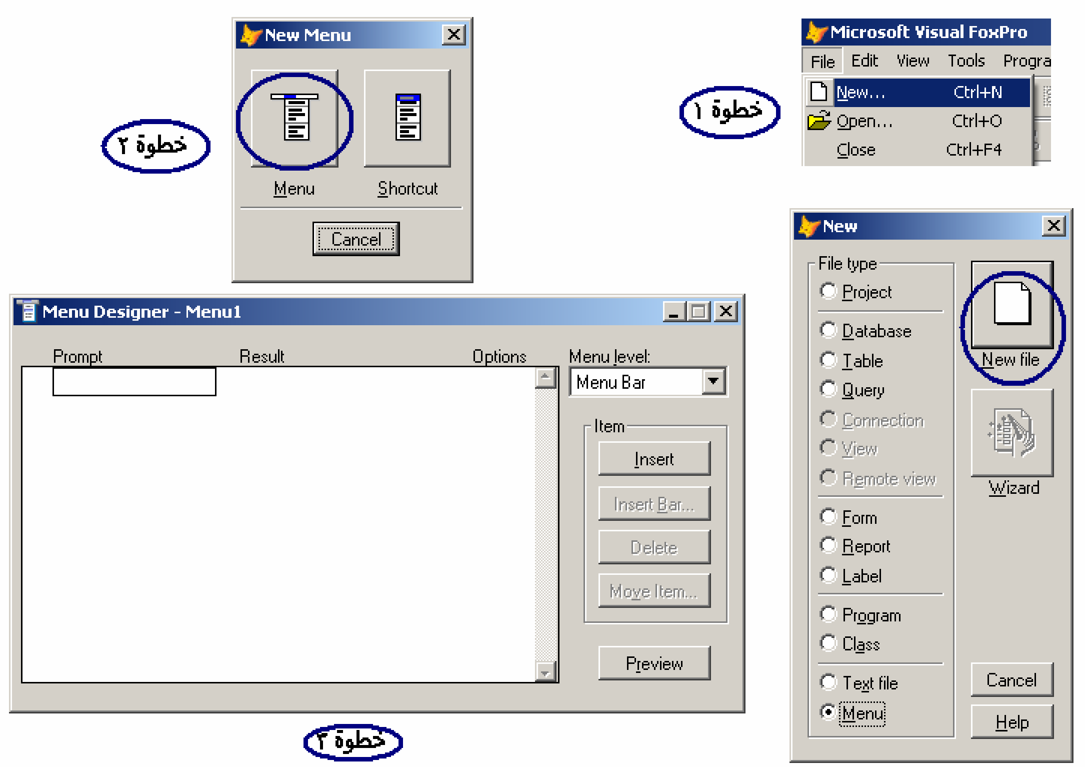
حيث mybar هو عبارة عن عنصر فى النموذج من النوع shapebox

وفى الشريط الخلفى الذى يمثل شريط القائمة يتم كتابة فى حدث تحرك الماوس

```
thisform.mybar.Visible = .f.
```

ويمكن استبدال الـ shape بعنصر من نوع container ونضبط صورة متكررة فيه ليكون الشكل اكثر جمالا.

مصمم القوائم :-

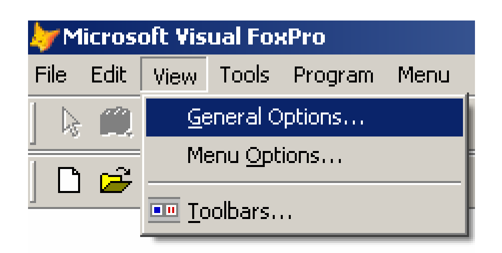


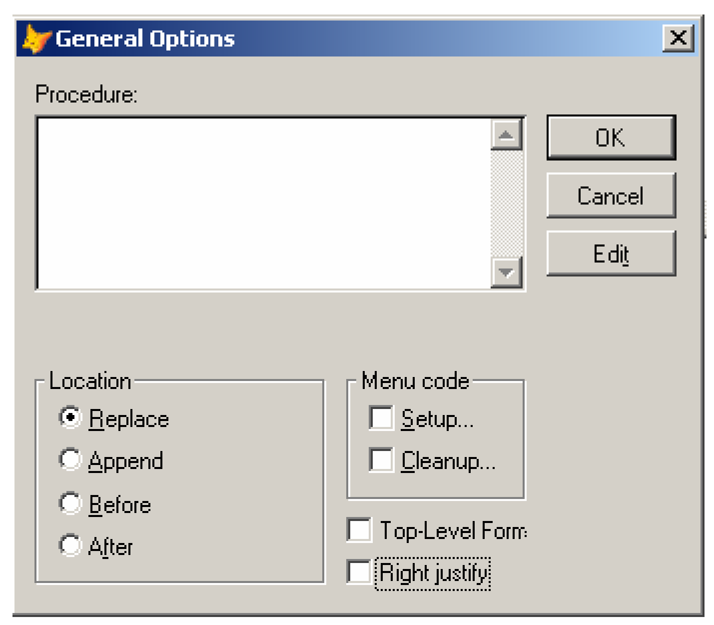
شكل ٦٧ : مصمم القوائم

Prompt : اسم العنصر
Result : نتيجة حدث اختيار العنصر من القائمة
٤ اختيارات command,submenu,padname & procedure
Options : خيارات للتحكم

وبعد تصميم القائمة يتم انتاج ملف تعليمات

وهناك خيارات هامة يمكن ضبطها قبل انتاج ملف التعليمات





شكل ٦٨ : خيارات هامة

Replace	تظهر القائمة بحيث تحل محل القائمة الموجودة
Append	يتم اضافة القائمة الى القوائم الموجودة
Before	تظهر قبل القوائم الموجودة
After	تظهر بعد القوائم الموجودة
Top-Level Form	قائمة خاصة بنموذج مستقل ويجب ان تاخذ ذلك فى الاعتبار جيدا (تحديد فى اى نموذج ستعمل القائمة)
Right Justify	ضبط القائمة لليمين
setup	اجراء ينفذ عند تحميل القائمة
cleanup	اجراء ينفذ بعد الانتهاء من مسح القائمة

ترجمة القائمة والحصول على التعليمات :-

يتم ذلك بسهولة عن طريق اختيار generate من القائمة Menu

استدعاء ملف القائمة الذى يحتوى على التعليمات الخاصة بها :-

استدعاء القائمة من نموذج مستقل

```
DO mainmenu.mpr with thisform,.t.
```

اذا كانت قائمة خاصة بواجهة فيجوال فوكس برو يكفى

```
DO mainmenu.mpr
```

حيث mainmenu.mpr هو الملف الذى تم انتاجه من مصمم القوائم عن طريق generate

الامر : set sysmenu to يلغى القائمة الموجودة بواجهة فيجوال فوكس برو
ولاعادتها مرة ثانية

```
SET SYSMENU TO default
```

ولجعلها جهة اليمين

```
SET SYSMENU TO rtljustify
```

ولاعادتها جهة اليسار

```
SET SYSMENU TO ltrjustify
```

ملحوظة اخيرة : اذا فتحت الملف المنتج بمصمم القوائم ويحوى الامتداد .mpr يمكنك ان ترى التعليمات المستخدمة لعمل القوائم



الباب السادس

ملفات المشاريع

مقدمة :

من المفترض انك قد تعلمت كيفية انشاء ملفات المشاريع والتعامل معها من خلال كتاب السيد محمد الهدهد فى دروس قواعد بيانات فيجوال فوكس برو والتى تم الاشارة اليه اكثر من مرة سابقا

لهذا سوف نركز الان على معلومة محددة خاصة بنقطة البداية فى ملف المشاريع والذى غالبا يكون عبارة عن ملف برنامج او اجراء PRG File.

اذا كان المشروع تحت واجهة فيجوال فوكس برو :

- ملف البداية باى اسم تختاره main.prg

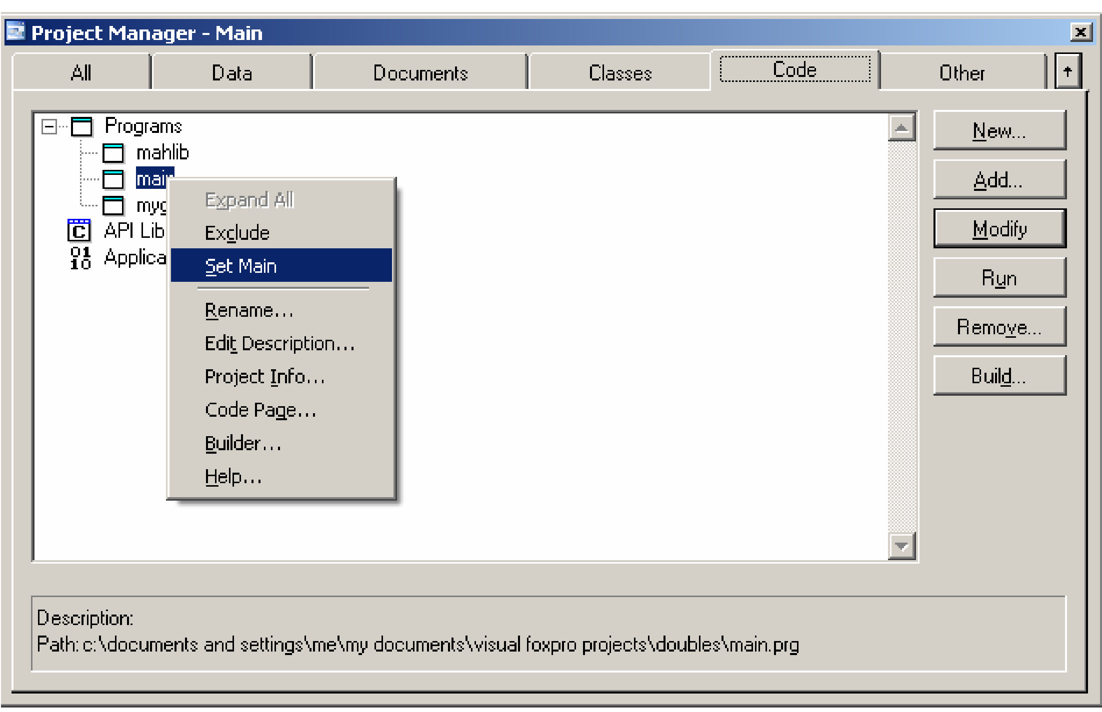
```
_screen.Caption = "my application name"
DO mymenu.mpr
SET SYSMENU TO rtljustify
DO mymainform.scx
READ events
```

وهكذا يمكنك تحديد اسم لبرنامجك بدلا من اسم فيجوال فوكس برو
وتستدعى القائمة الخاصة ببرنامجك
وتجعلها تظهر من اليمين لليسار
ثم تنادى النموذج الرئيسى لبرنامجك
ثم تنقل التحكم لهذا النموذج

اذا كان المشروع نموذج مستقل :

```
_screen.Visible = .f.
DO mymainform.scx
READ events
```

قم باخفاء واجهة فيجوال فوكس برو اولا
نادى النموذج الرئيسى فى البرنامج
انقل التحكم اليه



شكل ٦٩ : تحديد نقطة البداية فى المشروع

مشاريع جيدة للتعلم :

ياتى مع الكتاب عدد من المشاريع الجيدة للتعلم – بعض هذه المشاريع غير كامل – بعضها غير منظم – لانها كانت نتيجة عمل تجارب للتعلم – وليست مشاريع للاسواق.

مشاريع مفتوحة المصدر على الانترنت :

يوجد العديد من المشارع مفتوحة المصدر للتعلم والاستفادة على موقع www.sourceforge.net
ومن امثلتها احد المشاريع الذى قمت انا بعمله www.sourceforge.net/projects/doublesvsoop
وعند فتح هذا المشروع سوف تجد ان اللغة التى استعملت هى Visual FoxPro اضغط عليها لتحصل على كافة المشاريع التى تم تطويرها باللغة داخل الموقع

فيجوال فوكس برو ٩ :

هىاحدث اصدارات اللغة حتى تاريخ هذا الكتاب – ويحتوى على الكثير من التطويرات خاصة فى التقارير والجرافك وغيرها – وجميع امثلة الكتاب يمكن تطبيقها به او بالاصدارات القديمة مثل فيجوال فوكس برو ٦

كلمة الخاتمة :

ارجو من الله ان يكون هذا العمل صالحا وخالصا لوجهه الكريم – فلا تنسانا ياخى الحبيب من دعوة صالحة فى ظهر الغيب – ولا تتردد فى طرح اى سوال – او طلب اى مساعدة – ولا تكتفى ابدا بما تقرائه من اى كتاب سواء كان مرجعا كبيرا او كتيب صغير مثل هذا الكتاب وانما دائما واصل القراءة فى مرجع اللغة الاصلى الذى ياتى معها وتابع التغيرات على الانترنت وافتح المنتديات وتعرف على الاخرين لتفيد وتستفيد وكان الله فى عون الجميع.

والله الموفق

واخر دعوانا ان الحمد لله رب العالمبن .